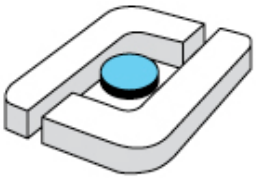


Hochschule Osnabrück
University of Applied Sciences

Fakultät Ingenieurwissenschaften und Informatik

Masterarbeit

über das Thema

Redesign of a PLC-controlled Kawasaki robot work cell and programming for assembly tasks

vorgelegt durch

Gonzalo Troyas García

11 Juni 2013

HOCHSCHULE OSNABRÜCK
Fakultät Ingenieurwissenschaften und Informatik

Masterarbeit

Thema: Redesign of a PLC-controlled Kawasaki robot work cell and programming for assembly tasks

für
Herrn
Gonzalo Troyas García

geboren am: 05.07.1989
in: Pamplona (Spanien)

Erstprüfer: Dr. Dirk Rokossa

Zweitprüfer: Dr. Klaus Kuhnke

Beginn: 15.02.2013

Abgabe: 11.06.2013

_____ Erstprüfer

_____ Master

_____ Zweitprüfer

Erklärung

Hiermit versichere ich, dass ich meine Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: 11.06.2013 .....
(Unterschrift)

ABSTRACT

This is a Master's thesis preformed by one student of mechanical engineering of the Public University of Navarra at the University of Applied Sciences of Osnabrück. In this work, a Kawasaki handling robot will be integrated into a Soft-PLC-controlled assembly line. Firstly, a redesign and improvement of the work cell is performed using 3D-CAD software (Catia V5R19). After, the robot is programmed in order to work as a server so that the PLC can control it totally. Furthermore, the control program from the Soft-PLC must be partially reprogrammed for this process. Finally, a real assembly operation is realized in order to verify that everything works properly. For this assembly operation some new tools must be designed and manufactured. The robot is programmed for this motion operation as well as a server side in order to execute this program automatically.

RESUMEN

Este el proyecto fin de carrera (PFC) realizado por un estudiante de ingeniería industrial (espacialidad en mecánica) de la Universidad Pública de Navarra en la Universidad de Ciencias Aplicadas de Osnabrück. Durante este trabajo, un robot de manipulación Kawasaki va a ser integrado dentro de una línea de montaje controlada por un sistema Soft-PLC. En primer lugar, un nuevo diseño de la célula de trabajo será realizado con ayuda de software 3D-CAD (Catia V5R19). A continuación, el robot será programado con el objetivo de trabajar como un servidor y ser controlado totalmente desde el PLC. Además, el programa de control del Soft-PLC será reprogramado para este proceso. Para finalizar, el ensamblaje de un producto real se llevará a cabo para verificar que todo funciona adecuadamente. Algunas herramientas nuevas deberán ser diseñadas y fabricadas para este ensamblaje. El robot será programado para este proceso teniendo en cuenta que este programa debe ser ejecutado automáticamente desde el PLC.

ZUSAMMENFASSUNG

Dies ist eine Masterarbeit eines Studenten des Maschinenbaus von der öffentlichen Universität von Navarra an der Hochschule Osnabrück. In dieser Arbeit wird ein Kawasaki Roboter in eine Soft-SPS-gesteuerte Fertigungslinie integriert werden. Zunächst wird eine Neugestaltung und Verbesserung der Arbeitszelle unter der Verwendung von 3D-CAD-Software (Catia V5R19) vorgenommen. Danach wird der Roboter programmiert um als Server zu arbeiten, so dass die SPS den Roboter vollständig kontrolliert. Darüber hinaus muss das Steuerungsprogramm der Soft-SPS für diesen Prozess teilweise neu programmiert werden. Anschließend wird ein Montagevorgang realisiert, um sicherzustellen, dass alles ordnungsgemäß funktioniert. Für diese Montage müssen einige neue Werkzeuge konstruiert und gefertigt werden. Der Roboter ist für diesen Bewegungsvorgang sowie für eine Server-Seite programmiert, um diese automatisch starten.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

It has come a long way since Henry Ford introduced the assembly line concept in 1901. Since then, this concept has been highly developed to reach the level of the current assembly lines. Automatic devices such as robots have replaced the workers. Nowadays, assembly lines even without a single worker can be found.

All assembly lines must be controlled somehow. This control must be based on an active and effective communication between the controller and all the automatic devices around it such as robots, sensors, conveyor belts, stoppers, etc.

The integration of a new robot into an assembly line carries a list of steps to fulfill with the aim of achieving a complete and effective integration. This thesis will show some of these required steps.

1.1. OBJECTIVES

Following the topic of the introduction, the main aim of this project is the integration of one robot into an existing assembly line. This integration will include the communication between the robot and the control system, the work station where this robot will work on, the development of some improvements at the work cell and a final assembly operation to show that everything works correctly. Each objective will be explained more in detail.

The final objective for the control system is the creation of only one client or PLC control program. In this way, each robot will act as a server and the PLC will control all these robots. This is not the objective of this master's thesis. For now, there will be a specific control program acting as the client and a specific server.

A server program is required so that the robot can communicate automatically with the central control (PLC) by using standardized protocols. The client control program also has to be completed in order to finish the previous work of other students. Furthermore, the work station has been almost mounted (all related to the mechanic and electronic) but some development and installations must be performed for full operation. Moreover, the work cell is not appropriately built for working at maximum speed. When the robot works at this speed, the whole cell moves doing the work inaccurately. This must be solved in order to work at maximum speed and to lower cycle times. Finally, there is no better way to show that everything works properly than an assembly operation controlling the robot and all the electronic and mechanic devices from the PLC. The PLC will control everything and the robot will just act as a server.

1.2. PROCEEDING

The first step required, as in the most of the projects, is to know the different technical characteristics, specifications and possibilities of the robot. This includes the way of

programming for both motion and communication processes, the way it communicates with the PC, the possibilities for this communication, etcetera.

Once that all these points are clarified, the next step is the creation of a server program in order to communicate automatically with the pre-set client program of the control of the assembly line. Before that, the client program of the PLC must be understood so that the server is correctly designed and programmed in order to answer what it might require. Different possibilities such as the used communication protocol are also discussed.

After all the electronic and mechanic devices at the work station are working and these are able to send/receive their signals to/from the control program, the electronic integration of the robot will be finished.

From here, the mechanic integration must be performed. The development of new parts for the work cell will be implemented by using Catia V5R19. This 3D CAD software will help with the design and this 3D design will be used to show some images of it in order to understand some points more easily. Everything will be manufactured and built and the cell will be located in the assembly line.

Finally, the assembly operation will be discussed and designed, designing everything needed for an accurate assembly operation including the new work tray and the new tools and resolving any problems that may arise during this process. With this assembly operation, all the work realized until then would be easily seen.

2. TECHNICAL BASES

2.1. PROGRAMMABLE LOGIC CONTROLLER

A *programmable logic controller* (PLC) is a special form of microprocessor-based controller that uses programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting, and arithmetic in order to control machines and processes (see figure 1). It is designed to be operated by engineers with perhaps a limited knowledge of computers and computing languages. Thus, the designers of the PLC have pre-programmed it so that the control program can be entered using a simple, rather intuitive form of language. The term logic is used because programming is primarily concerned with implementing logic and switching operations. Input devices (that is, sensors such as switches) and output devices (motors, valves, etc.) in the system being controlled are connected to the PLC. The operator then enters a sequence of instructions, a program, into the memory of the PLC. The controller then monitors the inputs and outputs according to this program and carries out the control rules for which it has been programmed (1).

PLCs have the great advantage that the same basic controller can be used with a wide range of control systems. To modify a control system and the rules that are to be used, all that is necessary is for an operator to key in a different set of instructions.

The first PLC was developed in 1969. PLCs are now widely used and extend from small, self-contained units for use with perhaps 20 digital inputs/outputs to modular systems that can be used for large numbers of inputs/outputs, handle digital or analog inputs/outputs, and carry out proportional-integral-derivative control modes (1).

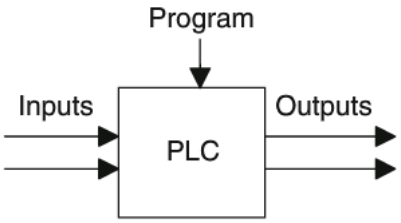


Figure 1: PLC (1)

2.1.1. HARDWARE-PLC

Typically, a PLC system has the basic functional components of processor unit, memory, power supply unit, input/output interface section, communications interface and the programming device. Figure 2 shows the basic arrangement (1).

- The *processor unit* or *central processing unit* (CPU) is the unit containing the microprocessor. This unit interprets the input signals and carries out the control actions according to the program stored in its memory, communicating the decisions as action signals to the outputs.

- The *power supply unit* is needed to convert the mains AC voltage to the low DC voltage (5 V) necessary for the processor and the circuits in the input and output interface modules.

- The *programming device* is used to enter the required program into the memory of the processor. The program is developed in the device and then transferred to the memory unit of the PLC.

- The *memory unit* is where the program containing the control actions to be exercised by the microprocessor is stored and where the data is stored from the input for processing and for the output.

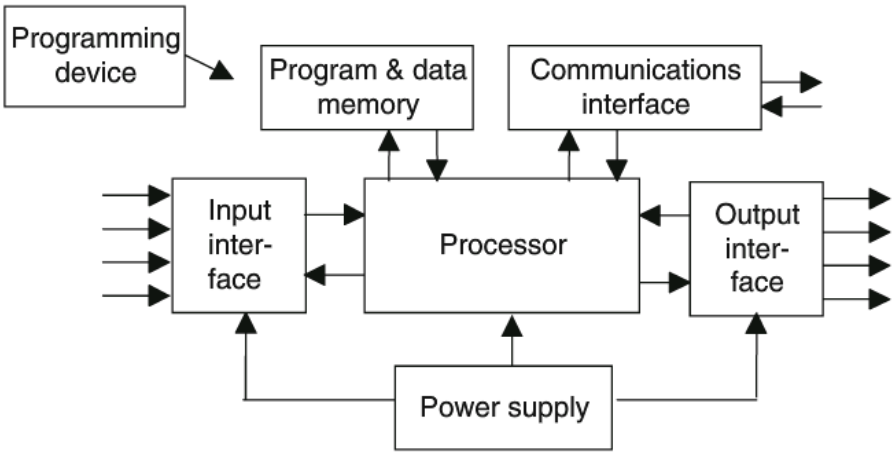


Figure 2: basic arrangement of a PLC (1)

- The *input and output sections* are where the processor receives information from external devices and communicates information to external devices. The inputs might thus be from switches or other sensors such as photoelectric cells, temperature sensors, flow sensors, or the like. The outputs might be to motor starter coils, solenoid valves, or similar things. Input and output devices can be classified as giving signals that are discrete, digital or analog.

- The *communications interface* is used to receive and transmit data on communication networks from or to other remote PLCs. It is concerned with such actions as device verification, data acquisition, synchronization between user applications and connection management.

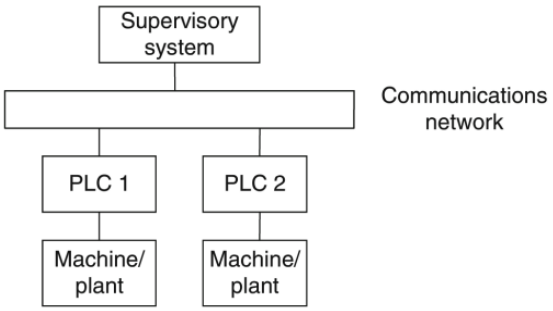


Figure 3: basic communication model (1)

2.1.2. SOFT-PLC

On the other hand, the second type of programmable logic controllers is known as “Soft-PLC”. Soft-PLC is a software product, which enabled the industry to build PLC's from standard computer and PLC hardware components. By using this Soft-PLC, users have the flexibility to select hardware according to their priorities: reliability, features, cost, performance, or vendor relationships. And it could be done independently for CPU and I/O, on a component basis. Soft-PLC was a liberating force for users of proprietary PLC's.

So, in contrast to a conventional hardware-based control, a Soft-PLC does not have its own hardware. It always has to be integrated in an existing computer system with a Windows operating system. Since a Soft-PLC runs within a Windows operating system, other Windows applications such as Excel or Access can access PLC-operands directly and fast without hardware wiring.

In contrast to a hardware-PLC the size of a Soft-PLC load memory can be changed flexibly (up to the maximum available memory size of the PC).

Since there are physical differences between a Hardware-PLC and a Windows computer, it is not possible to implement every feature of a Hardware-PLC in a Soft-PLC. However, existing programs could be executable on Soft-PLC with small changes (2).

2.1.3. PROGRAMMABLE LOGIC CONTROLLER IN THE LAB: OpenMHS.

OpenMHS is a Soft-PLC with integrated simulator. It was developed in the laboratory for handling technology and robotics at the Hochschule Osnabrück (“Labor für Handhabungstechnik und Robotik”) by Dipl.–Ing. Martin Nardmann. OpenMHS offers all the functions of a conventional industrial PLC. OpenMHS is even able to operate as a conventional PLC. Through the use of modern computers, it is possible to achieve very short cycle times. In addition, OpenMHS offers the possibility of using interfaces and creating simulation studies.

This Soft-PLC is used to control the assembly line facility and the industrial robots in the laboratory for handling technology and robotics (“Labor für Handhabungstechnik und Robotik”) at the Hochschule Osnabrück.

2.1.3.1. OPERATION MODES OF OpenMHS

OpenMHS supports the creation of models for different time-dependent problems. Thereby, by using Open MHS it is possible to solve tasks of simulation, data acquisition, control and regulation. For this purpose, there are different approaches for calculating the models stored in this software. The calculation of the models is based on time increments.

These time steps would be as

- discrete simulation, based on events, generated.
- continuous simulation, at constant intervals, generated.
- continuous simulation, in "real time" pulses, generated.

These different types of calculation can be mixed to some extent application.

The variable calculation models allow it to be used by OpenMHS for:

- Simulation of material flows (frequently event).
- Data Acquisition (always real time, combined with an event-driven).
- Data analysis, offline (constant time steps).
- Soft-PLC (real-time ever, combined with an event-driven).
- PLC simulation (constant time steps, combined with an event-driven).
- Simulation of a contract manufacturing often (frequently event-driven).

2.1.3.2. FUNCTIONING AS A SOFT PLC (REAL-TIME MODE)

The calculation in this mode is realized at equidistant points in time. The PLC waits between these points the required time until the next cycle. Each calculation involves always the next three steps:

- At first, inputs of the PLC can be queried and written into memory. This memory is called the process input image.
- The second step is the processing of the program. The program picks there exclusively the input values in memory. In this way it is ensured that the input values remain constant during the program run. The calculated output values are written to the process image output table.
- In the third step, the hardware outputs are assigned the values from memory.

Figure 4 clarifies the work field.

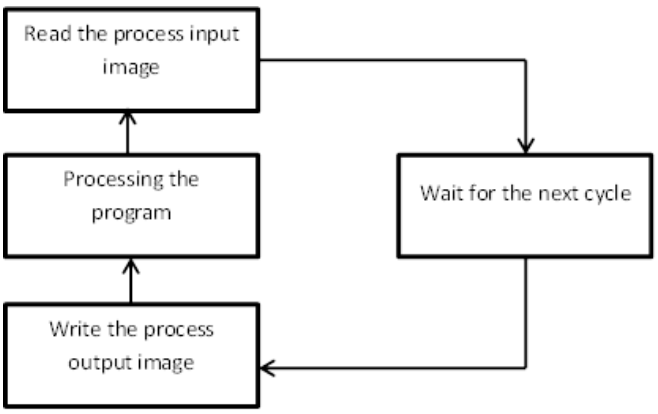


Figure 4: work cycle of a soft-PLC

2.1.3.3. STRUCTURE OF OpenMHS

The PLC control program of the assembly line is divided on some interfaces. The main interface shows one diagram of the whole system (see figure 5). Each sector with a work station is represented by a grey and orange figure whereas the corners are represented with a green and orange figure as it can be seen in figure 5. By clicking in each figure, a new interface opens where all the programming of each sector (both the work stations and the corners) is located. In chapter 5, the programming of sector 2 is explained more in detail.

In the upper left of figure 5, there is a grey rectangle. From here is where the transfer data to the RFID system is controlled. Nor this option nor the yellow rectangle are going to be used now so these are not going to be explained in detail. Inside the green rectangle titled "Freifahren" there are six orange figures. There is one of these figures for each sector where a work station is located (this means sectors 2, 3, 4, 7, 9, and 11). By clicking this figures a new interface opens showing the communication between the PLC and each robot (see figure 6). The most interesting programming inside this new interface is the programming of the handshake. The handshake between the robot in sector 2 and the PLC will be explained in detail in chapter 5. The other bytes are only screened in this interface.

Going back to the main interface shown in figure 5, it also contains a round button with the name "EIN". By clicking this button, the main motors of the assembly line will turn on moving the conveyor belt.

In the lower right side of figure 5, a green rectangle titled "Manuelle Einstellung RFID" is found. By clicking here, a new interface opens. From this new interface it is possible to change the initial data written into the RFID system. Form this interface it is from where the first destination of the tray and the first step to be performed are selected. The tray must be taken on top of a write head, located next to the PC where the Soft-PLC is installed, and this head will write into the transponder the code of the first destination sector and the first step to be performed in this sector. This will only be necessary to do the first time. Afterwards, the own robot will change this data and it will be also possible to do so from the next sector where the Kawasaki robot will send the tray to. The main area of this new interface is shown in figure 7.

In this figure 7, the destination sector can be chosen by turning the display to the number of the wanted sector. This will mean 20 for sector 2, 50 for sector 5, etcetera. By turning the second display, it is possible to choose the first step to be performed by the robot. As the assembly operation will have 5 different steps, the possibilities are from 1 to 6 meaning 6 that the process is already completed.

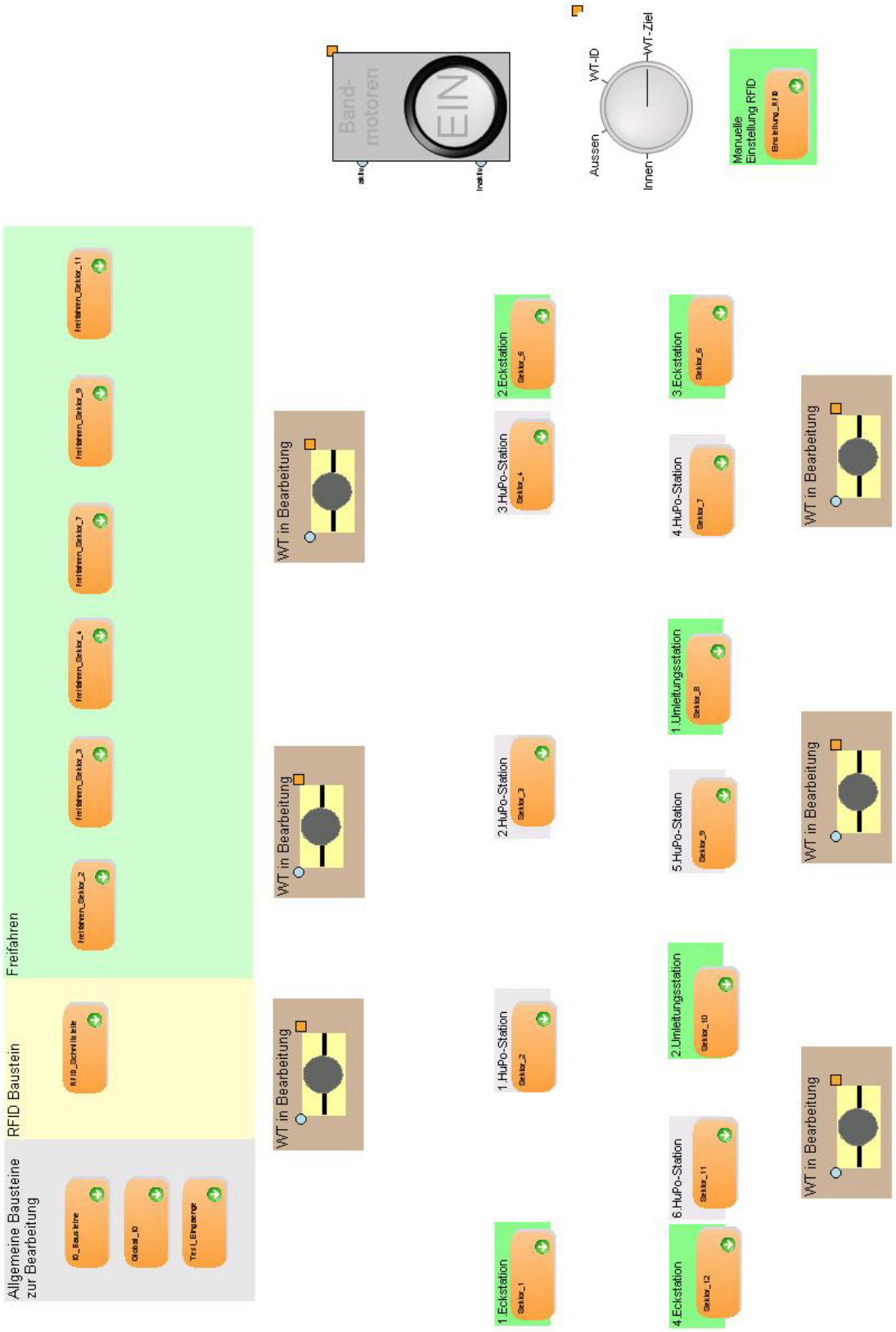


Figure 5: main interface panel of OpenMHS

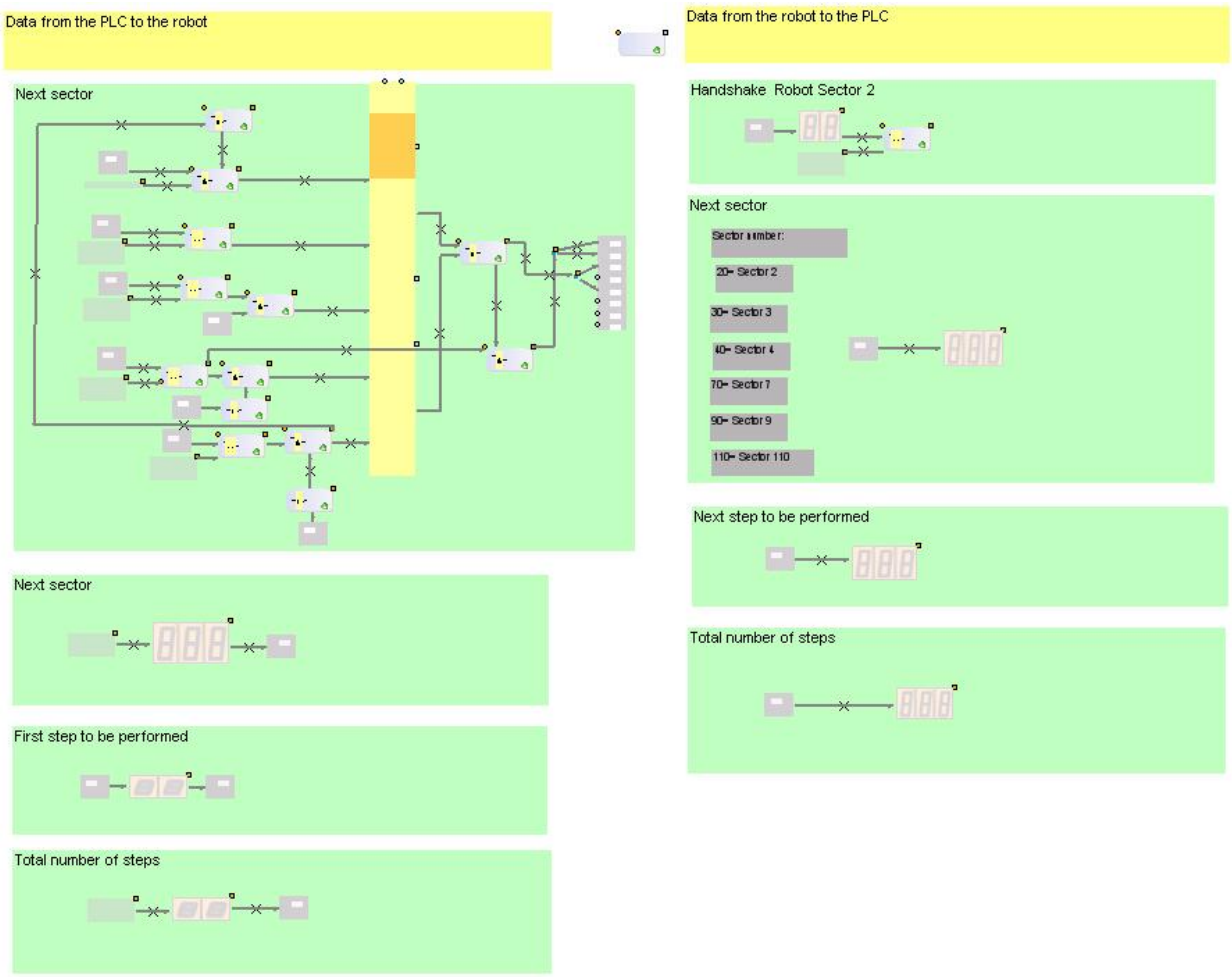


Figure 6: main interface panel of OpenMHS

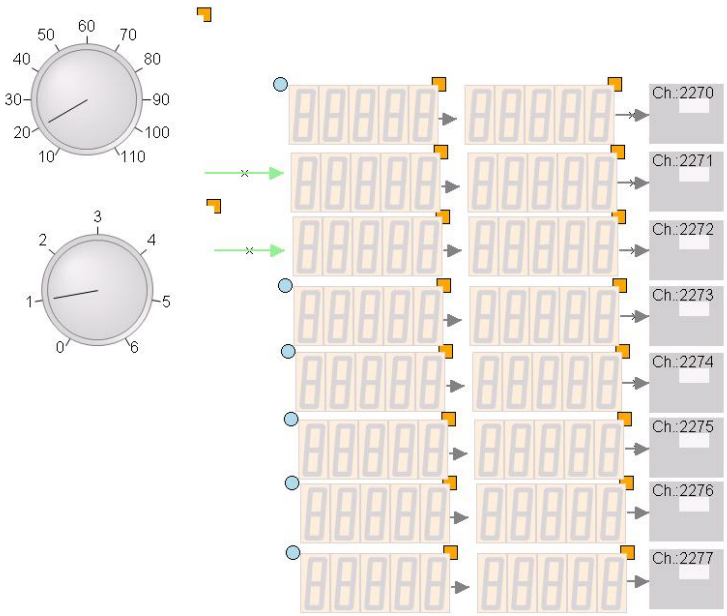


Figure 7: communication process interface

The second option for changing the destination of the tray and the first step to be performed will be when the robot raises the destination sector instructed by the robot. This destination will be instructed from the PC creen of the robot (PC located next to the robot and from which the robot is controlled) as it will be explained in chapter 5. In the future it could be done automatically by programming it inside the motion program. Once the tray gets to the destination instructed by the Kawasaki, the work station of this sector will go up and it will be possible to change this data as it is shown in the next figure 8. In figure 7, the modification of the data is done from sector 7. It works in the same way than the process in figure 7.

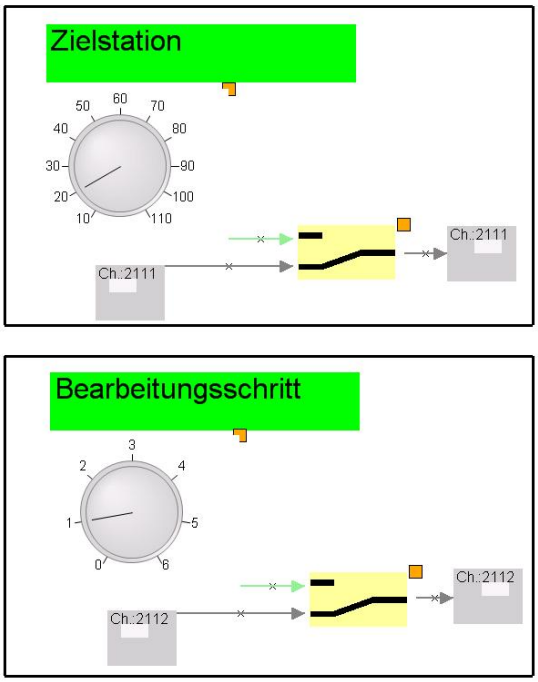


Figure 8: button for connection

Finally, figure 9 shows the menu from where the PLC program is executed and stopped. This menu is located on the left side of the window and if the program is wanted to be executed the button circled in green must be clicked. On the other hand, if the program is running and it is wanted to stop, the button circled in red must be clicked.

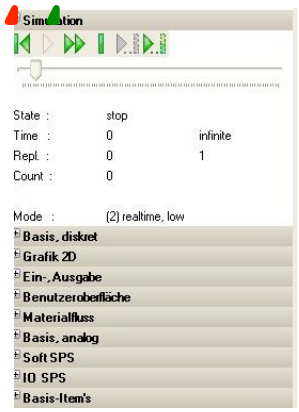


Figure 9: selection of sector interface

2.2. THE ROBOT: KAWASAKI R-SERIES RS05L

Kawasaki Heavy Industries was established as a ship builder in 1896. Today, the Kawasaki Heavy Industries (KHI) Group is comprised of over 100 companies in Japan and around the globe, together forming the world's leading industrial and technological business group (3).

The robot of this master's thesis is included inside the R-Series. Kawasaki's new R-Series Robots are setting the benchmark for all small to medium duty industrial robots. The compact design, along with industry leading speed, reach and work range make the R-Series Robots ideal for a wide range of applications throughout a multitude of diverse industries. These typical applications are from arc welding to dispensing; including sealing, material handling, material removal, machine tending, assembly, inspection and packaging between many others (3).

The robot is named "RS05L". It is the third "smallest" robot within the series. Its maximum payload is 5 kilogram. The new lightweight arm along with high-output high-revolution motors provide industry leading acceleration and high-speed operation. The acceleration rate automatically adjusts to suit the payload and robot posture to deliver optimum performance and the shortest cycle times. The slim arm design requires very little floor space. Multiple robots can be installed in "high-density" applications without impeding performance. Further specifications could be found in table 1 (3):

Table 1: specifications of Kawasaki RS05L (3; 4)

RS05L SPECIFICATIONS			
Type	Articulated		
Degrees of Freedom	6 axes		
Payload	5 kg		
Horizontal Reach	903 mm		
Vertical Reach	1,484 mm		
Repeatability	±0.03 mm		
Maximum Speed	9,300 mm/s		
Work Envelope (degrees) & Maximum Speed (degrees/s)	**Axis**	**Motion Range**	**Maximum Speed**
	JT1	±180°	300°/s
	JT2	+135° / -80°	300°/s
	JT3	+118° / -172°	300°/s
	JT4	±360°	460°/s
	JT5	±145°	460°/s
	JT6	±360°	740°/s
Wrist Load Capacity	**Axis**	**Maximum Torque**	**Moment of Inertia**
	JT4	12.3 N·m	0.4 kg·m^2
	JT5	12.3 N·m	0.4 kg·m^2
	JT6	7.0 N·m	0.12 kg·m^2
Motor(s)	Brushless AC Servomotor		
Brakes	All axes		

Hard Stop(s)	Adjustable mechanical stopper JT1	
Mass	37 kg (excluding Options)	
Body Colour	Kawasaki Standard	
Installation	Floor, wall, ceiling	
Environmental Conditions	Temperature	0 - 45°C
	Humidity	35 - 85%
	Vibration	Less than 0.5G
Protection Classification	Wrist: IP67 Base: IP65 *Equivalent	
Built-in Harness	Sensor harness 12 inputs, 24DC, GND	
Built-in Utilities	Pneumatic piping (Ø6 x 2 lines)	
Options	IP67 entire arm Riser (300/600 mm) Base plate Double/single solenoid valves (3 units max.) Air cleaning equipment (filter, regulator, mist separator)	
Controller	E77 (see E Controller data sheet for specifications)	

Kawasaki uses its own programming language named AS-Language. It consists of instructions formed by a keyword (this will be the name of the instruction or commands) followed by one space and at the end we will have the list of required parameters separated by comas. It will be shown better in the next lines.

Once the robot has been introduced, the next step would be to explain the communication between the two implicated sides: both the robot and the controller (PLC). This communication will be implemented with a socket communication function, which means that this communication function provides commands based on socket interface in TCP/IP communication enabling data communication between the robot controller and the PLC.

A *socket* is a software endpoint that establishes bidirectional communication between a server program and one or more client programs. The socket associates the server program with a specific hardware port on the machine where it runs so any client program anywhere in the network with a socket associated with that same port cans communicate with the server program (4).

A server program typically provides resources to a network of client programs. Client programs send requests to the server program, and the server program responds to the requests.

Kawasaki provides two protocols for the socket communication. On the one hand it provides UDP (User Datagram Protocol) and on the other TCP (Transmission Control Protocol).

Before describing the UDP and the TCP protocols in the next subchapters, it is necessary to explain what KRterm is. KRterm is terminal software for the Kawasaki robot controller. When this software is installed on a PC and the PC is connected to controller by Ethernet connection, the following operations are possible by TPC/IP communication: AS monitor command input from PC, saving/loading controller memory contents to/from PC, writing both motion and pc programs, etcetera (5).

2.2.1. PROGRAMMING

A *program* is a list of procedures that the robot will be made to do. When executing a program through the AS system, program steps (lines) are processed in order from top to bottom and the operations defined in each step are carried out by the robot (6).

The format of each step (line) of an AS language program is expressed in the following format:

Step number label program instruction ; comment

- **Step number:** a step number is automatically assigned to each line of a program. Steps are numbered consecutively beginning with 1 and are automatically renumbered whenever lines are inserted or deleted.

- **Label:** labels are used in a program to branch the program. A label can be either an integer form 1 to 9999 or a string of up 15 alphanumeric characters, periods or underscores (starting with an alphabetical character), followed by a colon (:). Labels are inserted at the beginning of a program line, right after the step number. Labels can be used as branch destinations from anywhere within the program.

- **Comment:** a semicolon (;) indicates that all the information to the right of it is a comment. Comments are not processed as program instructions when the program is executed, and are only used for explaining the program contents.

The program instructions are regularly executed in order from top to bottom of the program. This consecutive flow is changed when is an instruction such as GOTO or IF … GOTO. A CALL instruction calls up and executes a different program, but this does not change the order of the flow. When a RETURN instruction is executed, the processing returns to the caller program and resume form where it has left. This called program is named “subroutine”. By using the subroutine, it is possible to divide the program into a modular structure making the program much easier to understand (6).

2.2.1.1. PC PROGRAM

PC or process control programs are programs executed simultaneously with the robot control programs. PC programs are commonly used to control or monitor external devices by monitoring external I/O signals. The PC program and the robot control program can communicate with each other by using common variables or internal signals.

PC programs and robot control programs use instructions in common. Therefore, in some cases, a PC program can be executed as a robot control program. However, motion instructions other than BRAKE instruction cannot be used in PC programs.

A PC program can be set to start automatically when the controller power is turned ON. The steps are the following:

- Turn ON the system switch AUTOSTART.PC (or AUTOSTART2.PC – AUTOSTART5.PC).
- Create the program you want to start automatically and name it AUTOSTART.PC (or AUTOSTART2.PC – AUTOSTART5.PC).

Some monitor commands can be executed in programs by using program instruction MC. A PC program is executed by PCEXECUTE monitor command or by a program instruction that is executed from within a robot control program. PCABORT command can be used to stop execution of the PC program at any time. PCEND command ends the execution of the program after the current cycle is completed. PCCONTINUE command resumes execution of a program suspended by either PCABORT or because of an error (6).

2.2.1.2. ROBOT CONTROL PROGRAM

Robot control programs are programs that control the robot motion. All the program instructions including robot motion instructions might be used to create these programs.

There are main differences between the principal motion instructions. When an instruction begins with J (JMOVE, JAPPRO, etc.) it means that the movement will be performed in joint interpolated motion. On the other hand, if it starts with L (LMOVE, LAPPOR, etc.) the movement will be performed in linear interpolated motion. Explaining it another way, when the robot moves in joint interpolated motion, it moves so that the ratios of distance travelled to the total distance are equal at all joints throughout the movement from the starting pose to the end pose. And if it moves in linear interpolated motion, the origin form the tool coordinates (TCP) moves along a linear trajectory. The main motion instructions are going to be explained next (6):

- **JMOVE/LMOVE** *pose_variable, clap_number:* moves the robot to the specified pose (in joint or in linear interpolated motion).
- **JAPRRO/LAPPRO** *pose_variable, distance:* moves in tool Z direction to a specified distance from the taught pose (in either joint or linear interpolated motion).
- **JDEPART/LDEPART** *distance:* moves the robot to a pose at a specified distance away from the current pose along the Z axis of the tool coordinates (in either joint or linear interpolated motion).
- **HOME** *home_pose_number:* moves un joint interpolated motion to pose defined as HOME or HOME2. HOME or HOME2 should be defined beforehand using the SETHOME or SET2HOME commands.

- **DRAW/TDRAW** *X_translation, Y_translation, Z_translation, X_rotation, Y_rotation, Z_rotation, speed:* moves the robot in linear movement from the current pose and at the specified speed, the distance specified in the direction of the X, Y, Z axes and rotates the specified amount around each axis. DRAW instruction moves the robot based on the base coordinates whereas TDRAW instruction moves the robot based on the tool coordinates.

After the main motion instructions have been shown, the main instructions about speed and accuracy can be listed as well:

- **SPEED** *speed, rotational_speed, ALWAYS:* specifies the robot motion speed (the rotational speed is optional and usually specified in percentages between 0.01 and 100 (%). Also absolute speed can be set by specifying the speed with these units: DEG/S and DEG/MIN. If ALWAYS is entered, the speed set in this instruction remains valid until the next SPEED instruction is executed. If not entered, the speed is effective only for the next motion instruction.

- **ACCURACY** *distance ALWAYS FINE:* sets the accuracy when determining the robot pose. If ALWAYS is entered, the accuracy setting remains valid until the next ACCURACY instruction is executed. If FINE is entered, the robot pose is determined only when the current values match the taught pose regardless of the "distance" parameter setting.

2.2.2. INTERFACE PANEL SCREEN

Typically, an operation panel, known as interlock panel, is required to operate the robot and peripheral equipment together through a variety of hard switches and lamps. This controller provides an interface panel screen on the touch panel and enables the setting of the switches and lamps, changing arrangement of these on the screen. This subchapter describes the interface panel screen of the robot Kawasaki RS05L (7).

[I/F Panel] can be displayed on the pull-down menu in B area as shown in figure 10. Moving cursor to [I/F Panel] and pressing ENTER switches the B and C areas to the interface panel as shown in figure 11. Or, activate the B area, and press [I/F Screen Change] on the TP. [I/F Screen Change] switches between teach screen and interface panel screen each time the key is pressed.

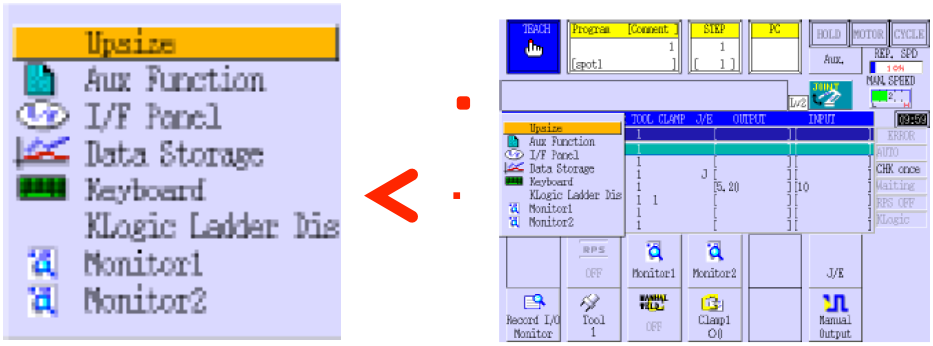


Figure 10: pull-down menu

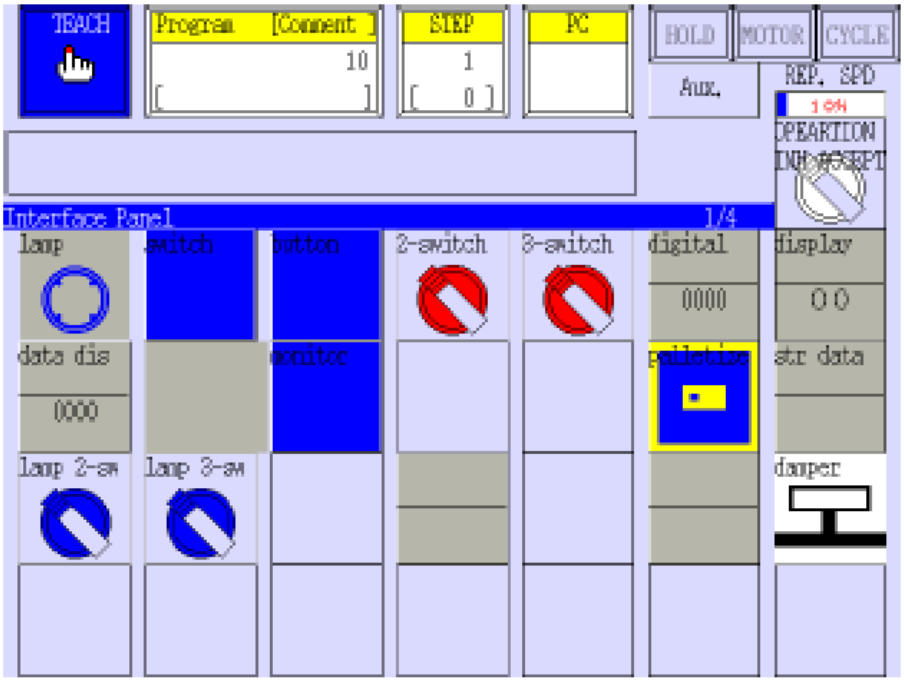


Figure 11: interface panel screen

An Interface panel consists of four pages or screens. Each screen can have up to 28 devices or displays (switches, lamps, etc.). Only devices that can be set in Aux. 0509 are available for the I/F Panel. Selecting Aux. 0509 displays the screen shown in figure 12. This screen also consists of four pages with the page number displayed on the above right of the screen. Pressing [Next Page] switches to the next screen.

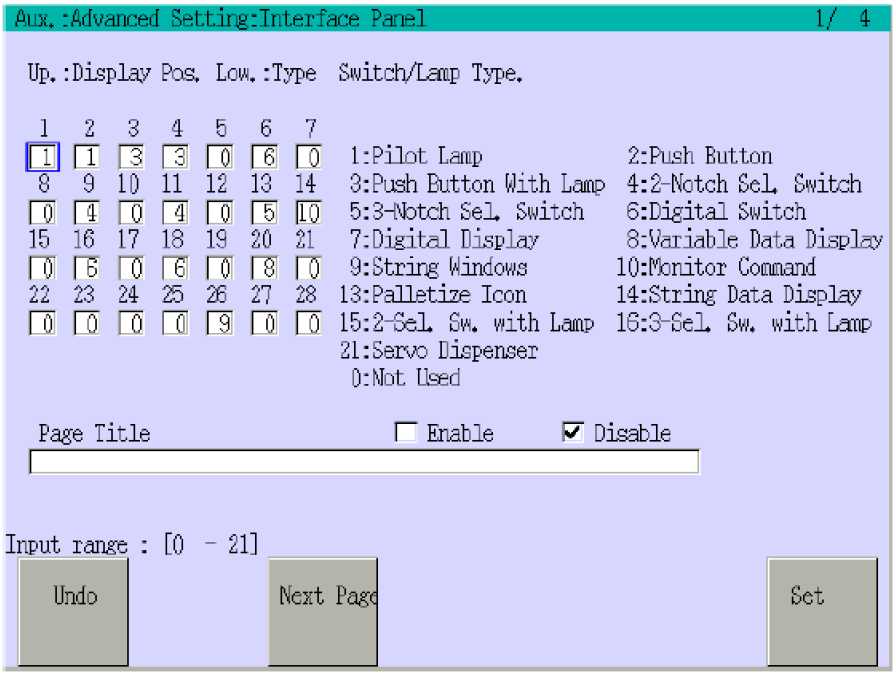


Figure 12: selection of device screen

Inputting a device type number from 1 to 10 and 13 to 16 under the device position number sets the type of the device. Inputting 0 makes the device position blank on the interface

panel. Once a device type number is input, pressing <Set> displays the device setting screen corresponding to its type.

On the next lines the function and procedure for setting devices provided in this controller will be described. The number before the input is the number that should be written to select each device (i.e. 1 for Pilot Lamp, 21 for Servo Dispenser) (7).

Table 2: list of colours (7)

No.	Color	No.	Color	No.	Color	No.	Color
0	Gray	4	Green	8	Pink	12	Navy
1	Blue	5	Pale Blue	9	White	13	Reddish Brown
2	Red	6	Yellow	10	Black	14	Dark Green
3	Orange	7	White	11	Cyan	15	Lavender

1. PILOT LAMP: lamp that will turn to the selected colour assigned to ON/OFF when the signal set in [Signal Number (Lamp)] is ON or OFF respectively.

2. PUSH BUTTON: pressing the switch turns ON the signal set in [Signal Number (Switch)]. When the same signal as the signal on the switch is ON, it automatically turns OFF if the interface panel screen is displayed or switched to the other screen or if the interface panel screen in the current page is switched to that of the other page. A colour can be set for both when it is ON and OFF.

3. PUSH BUTTON WITH LAMP: when the signal set in [Signal Number (Lamp)] is ON, the switch is coloured according to [Colour (ON)] regardless of the ON/OFF status of the switch. When the same signal as the set signal on the switch is ON, it automatically turns OFF if the interface panel screen is displayed or switched to the other screen or if the interface panel screen in the current page is switched to that of the other page.

4. 2-NOTCH SELECTOR SWITCH: two types of switching are available for 2-notch switch. Either [Up-Down] or [Rotation] in [Display Type] must be selected. A colour is set for each position. When turned the selector, the signal set in [Signal Number (Left), (Right), (Up) or (Down)] turns ON.

5. 3-NOTCH SELECTOR SWITCH: two types of switching are available for 3-notch switch. Either [Up-Middle-Down] or [Rotation] in [Display Type] must be selected. [Colour] sets the colour of the switch when it turned (Left), (Middle) and (Right) or (Up), (Middle) and (Down). When turned (position 1), the signal set in [Signal Number (position 1)] turns ON.

6. DIGITAL SWITCH: turns ON/OFF the signal assigned to each switch.

7. DIGITAL DISPLAY: shows on the screen the status of the required signals.

8. VARIABLE DATA DISPLAY: A variable will be displayed on the screen. In [Variable], input the variable name for the data to be displayed. In [Data Type], set how to display the data, by real numbers or integers.

9. STRING DISPLAY WINDOW: In [Window Number], input a number (1-8). Up to eight string display windows can be set per interface panel. Inputting 1 in [Window Size] specifies the window to have the standard width, inputting 2 doubles the width, and inputting 3 triples the width. There is a command from Kawasaki that once the window is created, it cans change the name, background colour, etc. It will be explained later.

10. MONITOR COMAND BUTTON: [Colour (ON) / (OFF)) sets the colour when this switch is pressed and released respectively. In [Command String], input the command to be executed. Moving cursor to [Command string] and pressing <Input> displays the keyboard to input the string. Up to 75 characters can be input.

13. PALLETIZE ICON: Pressing the palletize icon turns ON signal 1 and realizing the icon turns OFF the signal 1.

14. STRING DATA DISPLAY: in [Variable], input the string data to display. Up to ten characters can be input in each row of [Label 1 to 4].

15. 2-SELECTOR SWITCH WITH LAMP: two types of switching are available for 2-selector switch. Select either [Up – Down] or [Rotation] in [Display Type]. When the switch is turned (Left) and (Right) or (Up) and (Down), the signal set in [Signal Number (Left), (Right), (Up) or (Down)] turns ON. When the switch is turned (Left) and (Right) or (Up) and (Down), the signal set in [Signal Number (Left), (Right) or (Up), (Down)] turns ON.

16. 3-SELECTOR SWITCH WITH LAMP: it is the same as the 2-SELECTOR SWITCH DISPLAY WITH LAMP but in this case there are three switching positions (7; 14; 14; 14; 14; 14; 8).

Finally, some commands from Kawasaki own language AS-language could be used to write in the windows, name the labels and windows, etc. These commands can be executed from both process control programs or robot control programs and are (6):

1. IFPWPRINT *window_number, row, column, background_colour, label_colour ="character_string", "character string",…*

- It displays the specified character string in the string window set as told before.
- Window number: corresponds to the window number. Select from 1 to 8 (standard).
- Row: specifies the row in the selected window to display the string. Enter from 1 to 4.
- Column: specifies the column in the selected window to display the string. Enter from 1 to 70.

- Background_colour: selects the background colour from table 2.2.
- Label_colour: selects the colour of the characters displayed from table 2.2.
- Character_string: specifies the character string to display.
- Explanation: IFPWPRINT command can be used only when interface panel is available for use. If the parameters are not specified, the last setting of that particular window is selected.

2. IFPLABEL *position, "label_1", "label_2", "label_3", "label_4"*

- It sets and modifies the label of the icon at the specified position on the interface panel.
- Position: specifies the display position on the interface panel of the icon to set/modify the label. Set range: 1 to 112 where 1 is the first square in page 1 and 112 is the last in page 4.
- Labels: specifies the character string to display on the interface panel as the label of the specified icon.
- Explanation: sets and modifies the label for the icons displayed on the interface panel.

3. IFPTITLE *page_number, "title"*

- It sets and modifies the tittle for the specified page of the interface panel.
- Page_number: specifies the page of the interface panel to change the title. Setting range: 1 to 4.
- Title: specifies the character string to display on the page as the title "Interface Panel".

2.2.3. COMMUNICATION PROTOCOL VIA UDP/IP PROTOCOL

User datagram protocol, or UDP, is a simple, datagram-oriented, transport layer protocol: each output operation by a process produces exactly one UDP datagram, which causes one IP datagram to be sent. This is different from a stream-oriented protocol such as TCP where the amount of data written by an application may have little relationship to what actually gets sent in a single IP datagram.

UDP provides no reliability: it sends the datagrams that the application writes to the IP layer, but there is no guarantee that they ever reach their destination. Given this lack of reliability, it is tempted to think that the use of UPD protocol should be avoided and it should always be used a reliable protocol such as TCP (8).

There are only two UDP instructions: UDP_SENDTO (sends data) and UDP_RECVFROM (receives data). The instructions are as follow:

1. UDP_SENDTO *variable for returned value, IP address array variable, port number, character string variable array for sending data, number of elements, timeout*.

Its function is to send data based on UDP protocol. The data to be sent is specified in a character string variable array. The instruction creates socket, sends data and closes socket in one sequence. If communication error occurs, the error code is stored in the returned value storage variable and program execution does not stop.

2. UDP_RECVFROM *variable for returned value, port number, character string variable array for received data, number of elements, timeout, IP address array for received data, number of elements, timeout, IP address array variable, maximum number of bytes.*

It receives and stores data in the character string variable array based on UDP protocol. This instruction creates socket, receives data and closes socket in one sequence. If a communication error occurs, the error code is stored in the returned value storage variable, and program execution does no stop (5).

2.2.4. COMUNICATION PROTOCOL VIA TCP/IP.

Even though TCP and UDP protocols use the same network layer (IP), The transmission control protocol or TCP provides a totally different service to the application layer than UDP does. TCP provides a connection-oriented, reliable, byte stream service.

The term connection-oriented means the two applications using TCP (normally considered a client and a server) must establish a TCP connection with each other before they can exchange data. The typical analogy is dialling a telephone number, waiting for the other part to answer the phone and say "hello," and then saying who's calling. There are exactly two end points communicating with each other on a TCP connection.

TCP provides reliability by doing the following (8):

- The application data is broken into what TCP considers the best sized chunks to send. This is totally different from UDP, where each write by the application generates a UDP datagram of that size. The unit of information passed by TCP to IP is called a segment.

- When TCP sends a segment it maintains a timer, waiting for the other end to acknowledge reception of the segment. If an acknowledgment is not received in time, the segment is retransmitted.

- When TCP receives data from the other end of the connection, it sends an acknowledgment. This acknowledgment is not sent immediately, but normally delayed a fraction of a second.

- TCP maintains a checksum on its header and data. This is an end-to-end checksum whose purpose is to detect any modification of the data in transit. If a segment arrives with an invalid checksum, TCP discards it and does not acknowledge receiving it (it expects the sender to time out and retransmits.)

- Since TCP segments are transmitted as IP datagrams, and since IP datagrams can arrive out of order, TCP segments can arrive out of order. A receiving TCP re-

sequences the data if necessary, passing the received data in the correct order to the application.

- TCP also provides flow control. Each end of a TCP connection has a finite amount of buffer space. A receiving TCP only allows the other end to send as much data as the receiver has buffers for. This prevents a fast host from taking all the buffers on a slower host.

A stream of 8-bit bytes is exchanged across the TCP connection between the two applications. There are no record markers automatically inserted by TCP. This is what we called a byte stream service. If the application on one end writes 10 bytes, followed by a write of 20 bytes, followed by a write of 50 bytes, the application at the other end of the connection cannot tell what size the individual writes were. The other end may read the 80 bytes in four reads of 20 bytes at a time. One end puts a stream of bytes into TCP and the same, identical stream of bytes appears at the other end.

Also, TCP does not interpret the contents of the bytes at all. TCP has no idea if the data bytes being exchanged are binary data, ASCII characters, EBCDIC characters, and etcetera. The interpretation of this byte stream is up to the applications on each end of the connection (8).

In TCP communication there are seven instructions to perform the communication between the controller and the PLC. These eight instructions are TCP_LISTEN, TCP_ACCEPT, TCP_CONNECT, TCP_SEND, TCP_RECV, TCP_CLOSE, TCP_END_LISTEN and TCP_STATUS (5).

1. TCP_LISTEN *variable for returned value, port number*

This program instruction starts waiting for connection request (used by server-side to start communication service). Creates socket, binds it to the specified port number, and waits for connection request to that socket. If communication error occurs during execution, the error code is stored in the returned value storage variable and program execution does not stop.

2. TCP_ACCEPT *variable for returned value, port number, timeout, client IP address array variable*

Program instruction to check the connection request (used by server-side to start communication service). Checks if the connection request for socket communication has been received by the specified port, and if it has, establishes connection. Connection is completed when this instruction terminates normally. If communication error occurs during execution, the error code is stored in the specified returned value storage variable and program execution does not stop.

3. TCP_CONNECT *variable for returned value, port number, server IP address array variable, timeout*

Program instruction to request the connection (used by client-side to start communication service). Creates socket and binds to the specified port number. Then, connection request is

sent to the specified node, and connection is established. The node is determined by the IP address of the server. If communication error occurs during execution, the error code is stored in the returned value storage variable and program execution does no stop.

4. TCP_SEND *variable for returned value, socket ID number, character string variable array for sending data, number of elements, timeout*

Program instruction to send data. Sends data based on TCP protocol. The data to be sent is specified as character string variable array. If communication error occurs, the error code is stored in the returned value storage variable and program execution does not stop.

5. TCP_RECV *variable for returned value, socket ID number, character string variable array for received data, number of elements, timeout, maximum number of characters*

Program instruction to receive data. Receives data on TCP protocol, and stores it in the specified character string variable array. If communication error occurs, the error code is stored in the returned value storage variable and program execution does not stop.

6. TCP_CLOSE *variable for returned value, socket ID number*

Program instruction to abort/finish connection (used to terminate communication service). Cuts off connection for socket communication and closes socket. If communication error occurs, the error code is stored in the returned value storage variable, and program execution does not stop.

7. TCP_END_LISTEN *variable for returned value, port number*

Program instruction to abort connection. Ends waiting for connection request on the socket specified by TCP_LISTEN and close that socket. If communication error occurs, the error code is stored in the returned value storage variable, and program execution does not stop.

8. TCP_STATUS *variable for returned value, port number, socket-ID-array, error-code-array, sub-error-code-array, IP address array*

Its function is to store to the specified array variables, the status of the sockets used in TCP communication instructions. Data are stored in order of sockets ID number, as controlled within the robot controller. The data with the same array number in each parameter are data from same socket. When there is an error in the parameter, the program execution stops.

Due to the many advantages that TCP has over UDP, TCP/IP protocol will be used in the communication between the Kawasaki robot and the PLC. An example of this communication is shown in the next figure 13. The robot is located as the server side and all the TCP instructions are used without any parameters.

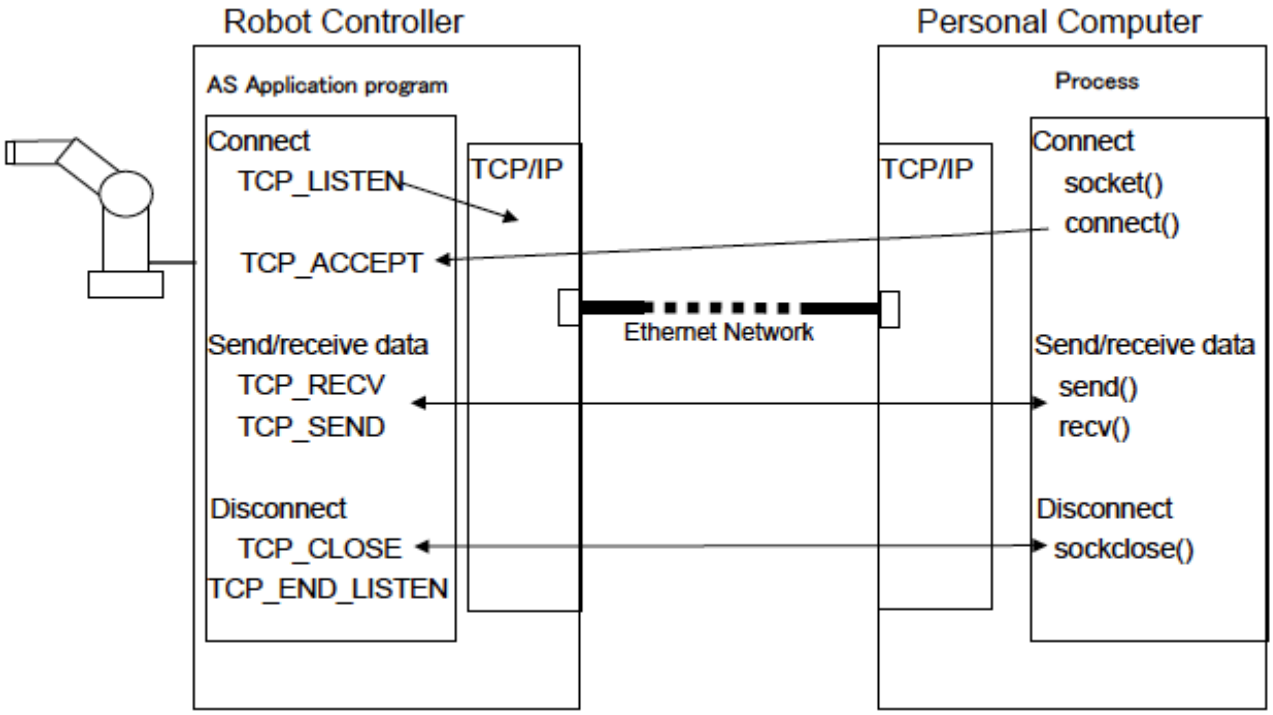


Figure 13: Kawasaki as server side (5)

2.3. RADIO FREQUENCY IDENTIFICATION (RFID)

In RFID systems, data is stored on an electronic data-carrying device: the transponder. However, unlike the smart card, the power supply to the data-carrying device and the data exchange between the data-carrying device and the reader are achieved without the use of galvanic contacts, using instead magnetic or electromagnetic fields. The underlying technical procedure is drawn from the fields of radio and radar engineering. Due to the numerous advantages of RFID systems compared with other identification systems, RFID systems are now beginning to conquer new mass markets. One example is the use of contactless smart cards as tickets for short-distance public transport (9).

An RFID system is always made up of two components (see figure 14):

- The *transponder*, which is located on the object to be identified.
- The interrogator or *reader*, which, depending upon the design and the technology used, may be a read or write/read device.

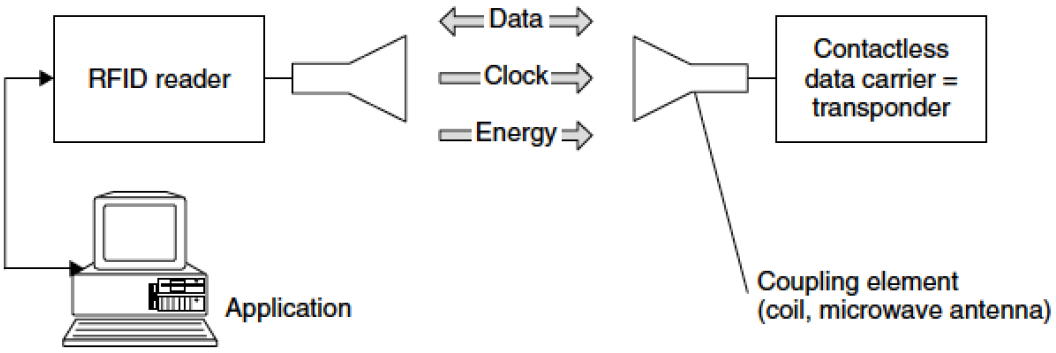


Figure 14: main components of a RFID system (9)

A reader typically contains a radio frequency module (transmitter and receiver), a control unit and coupling element to the transponder. In addition, many readers are fitted with an additional interface (RS 232, RS 485, etc.) to enable them to forward the data received to another system (PC, robot control system, etc.).

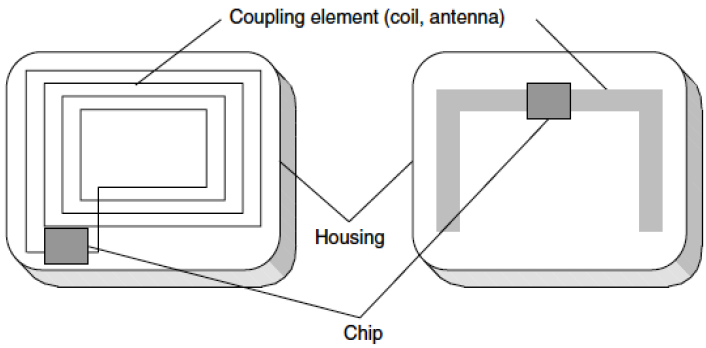


Figure 15: inductively coupled transponder with antenna coil (left), microwave transponder with dipolar antenna (right) (9)

The transponder, which represents the actual data-carrying device of an RFID system, normally consists of a coupling element and an electronic microchip (see figure 15).

When the transponder, which does not usually possess its own voltage supply (battery), is not within the interrogation zone of a reader it is totally passive. The transponder is only activated when it is with in the interrogation zone of a reader. The power required to activate the transponder is supplied to the transponder through the coupling unit (contactless), as are the timing pulse and data (9).

The most important differentiation criteria for RFID systems are the operating frequency of the reader, the physical coupling method and the range of the system. RFID systems are operated at widely differing frequencies, ranging from 135 kHz long waves to 5.8 GHz in the microwave range. Electric, magnetic and electromagnetic fields are used for the physical coupling. Finally, the achievable range of the system varies from a few millimetres to above 15 m.

RFID systems with a very small range, typically in the region of up to 1 cm, are known as close coupling systems. For operation, the transponder must either be inserted into the reader or positioned upon a surface provided for this purpose. Close coupling systems are coupled using both electric and magnetic fields and can theoretically be operated at any desired frequency between DC and 30 MHz because the operation of the transponder does not rely upon the radiation of fields.

Systems with write and read ranges of up to 1 m are known by the collective term of remote coupling systems. Almost all remote coupled systems are based upon an inductive (magnetic) coupling between reader and transponder. These systems are therefore also known as inductive radio systems. In addition there are also a few systems with capacitive (electric) coupling. At least 90% of all RFID systems currently sold are inductively coupled systems. For this reason there is now an enormous number of such systems on the market. There is also a series of standards that specify the technical parameters of transponder and reader for various standard applications. Frequencies below 135 kHz or 13.56 MHz are used as transmission frequencies. Some special applications are also operated at 27.125 MHz.

RFID systems with ranges significantly above 1 metre are known as long-range systems. All long-range systems operate using electromagnetic waves in the UHF and microwave range. The vast majority of such systems are also known as backscatter systems due to their physical operating principle. In addition, there are also long-range systems using surface acoustic wave transponders in the microwave range. All these systems are operated at the UHF frequencies of 868 MHz (Europe) and 915 MHz (USA) and at the microwave frequencies of 2.5 GHz and 5.8 GHz (9).

2.3.1. RFID IN THE LABORATORY

In the laboratory for handling technology and robotics (“Labor für Handhabungstechnik und Robotik”) an assembly line system can be found. It transports carrier trays along the line.

This assembly line system was improved in the summer semester of 2012 with an RFID system, consisting of 12 writing and reading stations and carrier trays equipped with transponders. Using this technique, it allows different control strategies for the carrier trays. For example, an automated product assembly process could be realized (10; 11).

The working frequency found in the system in the lab is 13.56 MHz. In this frequency range stamped or pressed coils can be used, which make the tags more cost-effective. It is used frequently, as in this case, in logistics applications, asset logging and certain factory applications. Due to the low costs, this frequency is ideal for applications where many read/write tags are used. The read/write tags cannot be embedded in metal and are not suitable for tool identification. They are not only cost-effective, but also due to the high frequency are 3 to 4 faster than the low frequency versions (11).

2.3.1.1. READ/WRITE HEAD

The read/write heads are located on each of the work stations of each robot and before the corners or the up/down changing lines. The model number form the read/write head in the lab is IQH1-FP-V1 from the company Pepperl + Fuchs, GmbH. The main features of this head are (11):

- Operating frequency 13.56 MHz.
- Conformal coated with ISO 15693.
- Transfer rate of 26 Kbit/s
- Reading/Writing distance: from 0 to 130 mm.
- Dual-LED for function display.

2.3.1.2 READ/WRITE TAG

The read/write tags are located under each carrier tray. Its model number is IQC33-50 from Pepperl + Fuchs as well. The main features of this tag are (11):

- Operating frequency 13.56 MHz.
- 16 Kbit available memory.
- 64-bit fix code.
- Readable and writable from both sides.
- Transfer rate of 26 Kbit/s.

3. REDESIGN OF THE KAWASAKI WORK CELL

3.1. DESIGN IN CATIA 3D

The Kawasaki robot in the laboratory is located in a work cell. This work cell is constructed of aluminium profiles screwed together by bolts and steel joints. This cell has one work table made of wood so the robot can work on both the assembly line and on this work table. As this table could be useful in the future it was decided to leave it as it was before. A 3D image of this cell can be seen in figure 16. The only parts missing in these images are the unions between the cell and the floor, but shown in figure 17, and the old structure of the robot shown in figure 18 with an adequate precision.

Even so, the whole structure was not sufficiently rigid. The first main problem could be found in the cell-floor union. An image of this union can be found in figure 17. This union made the whole structure not rigid enough and when the robot worked at working speed, the entire structure moved and waved making it impossible to work with an adequate precision.

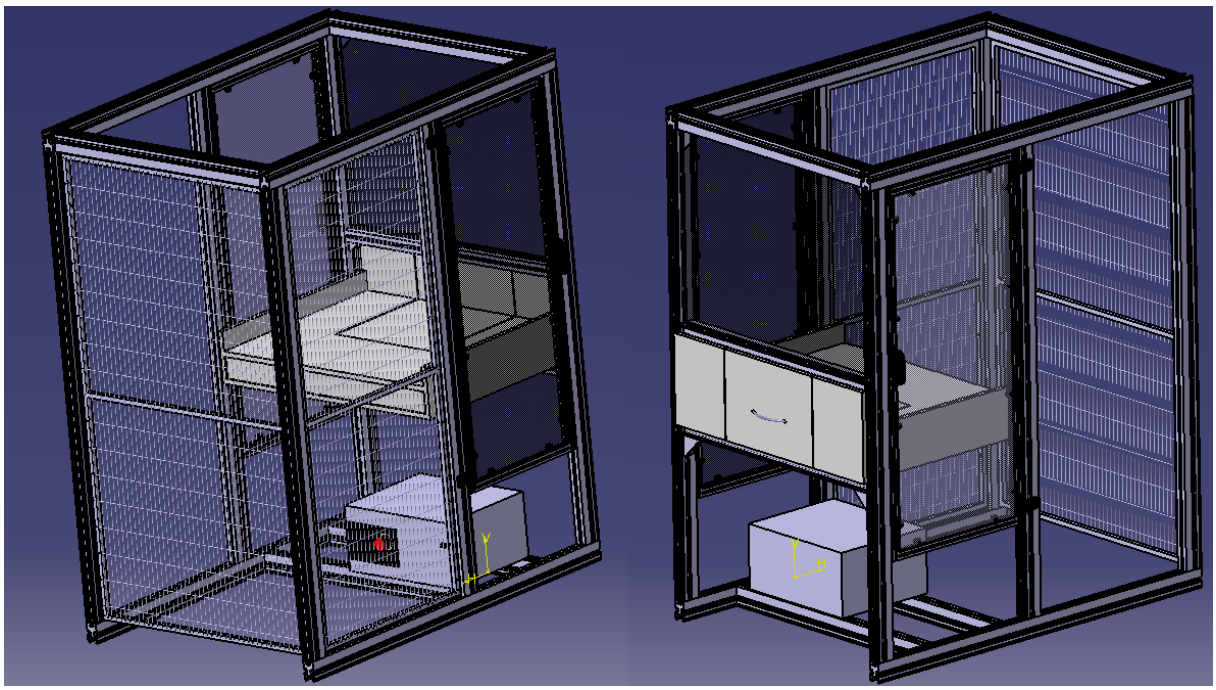


Figure 16: old cell of the robot

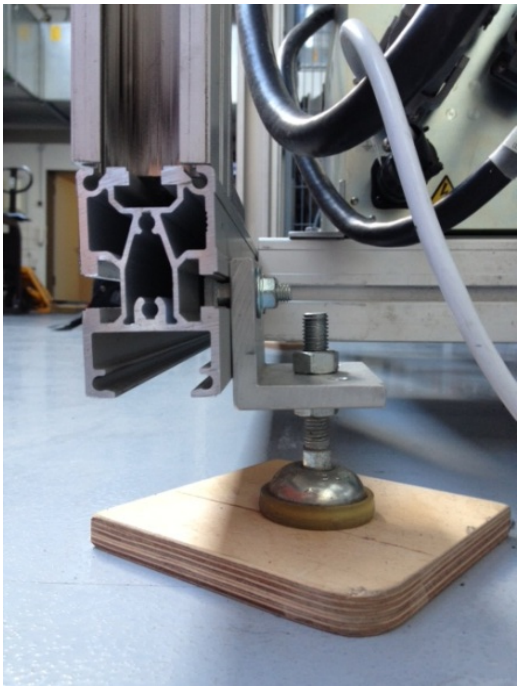


Figure 17: old union cell-floor

Figure 18: old structure of the robot

The second main problem, on the other hand, was located in the structure where the Kawasaki was installed (see figure 18). It was also constructed of aluminium profiles and fixed to the aluminium profiles of the cell. Furthermore, this union was made with bolts and was not rigid enough. When the Kawasaki moved slowly everything worked perfectly, but when it moved at the working speed, the whole structure moved violently moving the whole cell.

Thus, there were two main problems to be solved: the union cell-floor and the union base of Kawasaki-cell.

3.1.1. NEW PLATFORM FOR THE CELL

The solution for the first problem was the design and manufacturing of a new platform for the whole cell. There were already some examples existing in the laboratory so these examples were taken as the first idea. Afterwards, only a few changes were made in order to shape these examples to the Kawasaki solution.

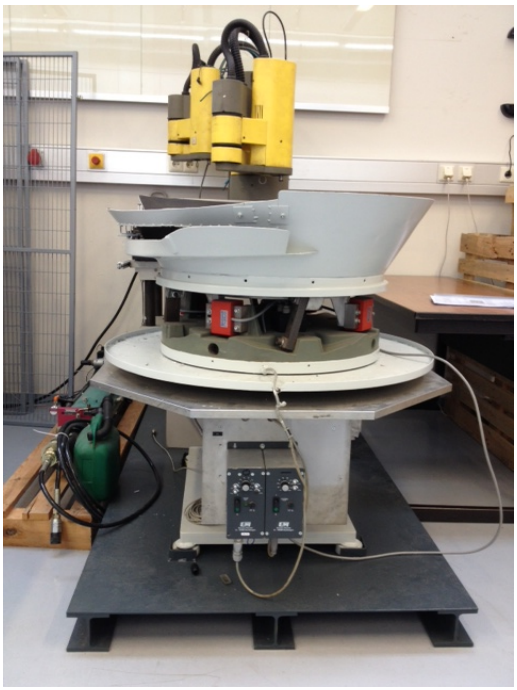


Figure 19: example of another platform

Figure 19 shows the platform of one of the robots in the laboratory. As it can be seen, this example is constructed mostly of a big steel plate and two or three double-T profiles welded to this plate. These structures are constructed in this way in order to be easily transported inside the laboratory, so its location could be changed without further problem by using a hand pallet truck.

The base of the cell of the Kawasaki was a rectangle of 1600 mm x 1095 mm. Thus, the idea of a steel plate of these dimensions came at first. In most of the examples in the laboratory, the solution was to manufacture a plate with different thicknesses. The thickness just under the robot is thicker than the thickness on the other sides so it has more stability where it is needed. Unfortunately, the shape of the cell of the Kawasaki robot made it impossible to manufacture a multi-thickness plate. If there were different thicknesses, the aluminium profiles would not contact the plate throughout its length. It would make the whole platform quite more expensive but it would gain the needed stability.

For this reason, the thickness of this new plate had to be equal all over its area. The next point was to think about this thickness. Even though the thickness in the first design was 20 mm, in the end it was changed to 15 mm. A thickness of 20 mm should give the weight and the stability needed to work at maximum speed. For an economic reason of price, and because it was thought that 15 mm would be enough, the thickness was changed to 15 mm. The final price of the plate would be three quarters of the original one of 20 mm. Figure 20 shows a 2D image of this plate. For further technical information see plan 3.1 in annex 2.

As it can also be seen in figure 20, there are some holes drilled through the plate. The ones at the sides (eight in total) are for the union between the plate and the cell. These have a diameter of 10 mm. The union item itself between the platform and the cell will be explained later. On the other hand, the four holes in the middle of the platform are for the union between the structure of the Kawasaki and the platform. These are not centred as might be supposed. The reason will be explained when the structure for the Kawasaki is seen.

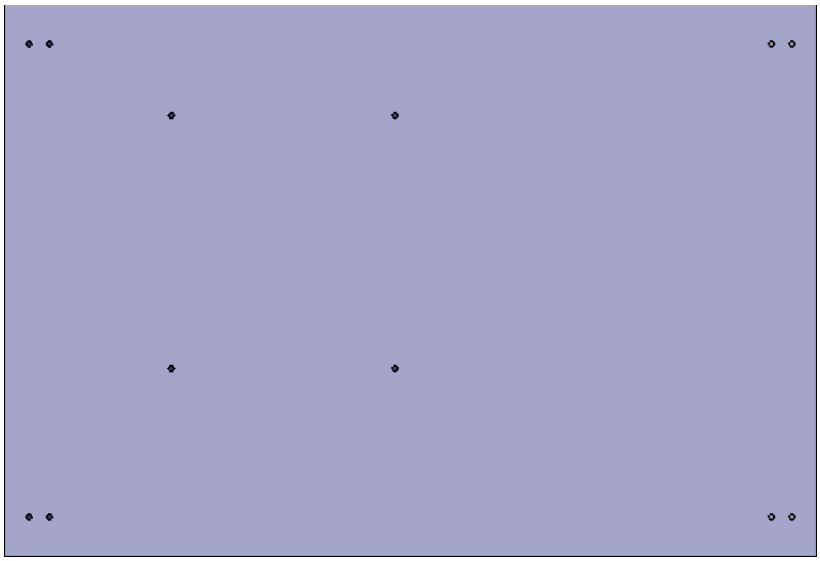


Figure 20: plate of the platform

The size of the double-T profiles should be discussed just after the plate because these are the two components of the platform. The profiles of the other bases were double-T profiles of height 100 or IPE-100. As they were suitable and there was the chance to get these in the Shop Floor, it was decided to build the platform with the same ones. There would be needed at least two but the weight of the whole structure led to the decision of manufacturing it with three. It was decided to locate them along the larger side of the plate so its final length would be 1600 mm so that the weight of them would be greater and it would give more stability to the whole structure. The profile in the middle did not need any hole but the ones at the sides needed the same holes as the plate in order to locate the bolts into them for the union between the cell and the whole platform. The diameter of these holes would be as well of 10 mm. As these profiles are symmetric the two profiles at the sides would be the same ones, just turning them around 180 degrees. Figure 21 and will show these three double-T profiles. For further technical information see plan 3.2 and 3.3 in annex 2.

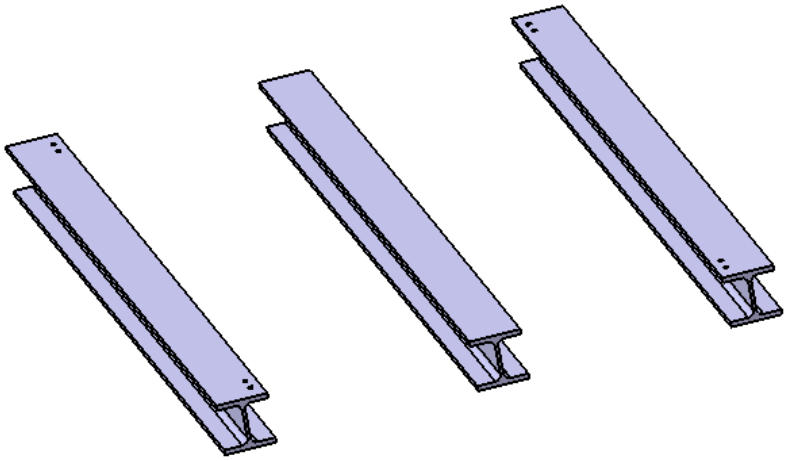


Figure 21: 3 double-T profiles

In plan 3.4 in annex 2 the whole platform including the plate and the three double-T profiles are shown. A 3D image of the platform can be seen as well in figure 22.

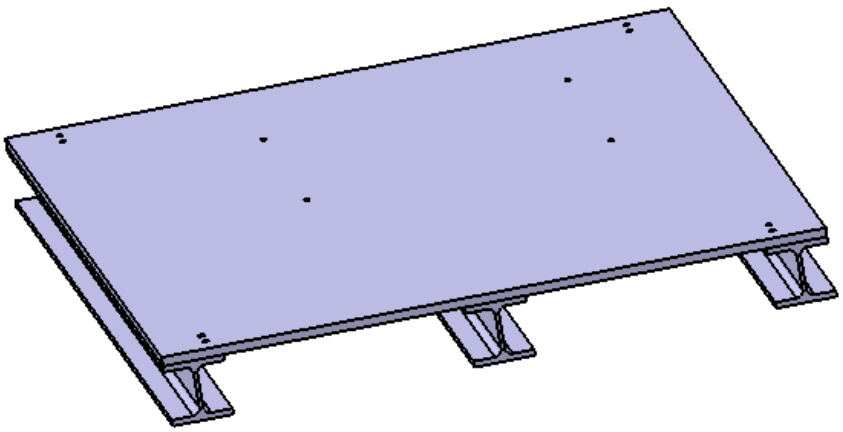


Figure 22: new platform

Furthermore, the mass of the whole platform including the three double-T profiles and the steel plate would be about 245 kg (if 7850 kg/m^3 is taken as the steel density and 8,1 kg/m as the linear density of a double-T profile IPE-100) and if the weight of the cell, the base of the Kawasaki and the weight of the own robot are added, it would amount between 300 and 350 kg. This mass should be enough to give stability to the whole cell preventing the unwanted movement.

As the structure of the Kawasaki is united directly to the plate of the platform, there is not physical connection between the cell and the robot itself except this big steel plate at the platform. Therefore, there was not a huge need of a solid union between these. The final decision was the manufacturing of the cheapest solution, which was the manufacturing of four L-profiles in order to screw the cell to the platform. The platform was designed with specific holes for screwing those bolts: two at the bottom side and two on the lateral side of

each L-profile (plan 3.5 in annex 2). A 3D detail of these L-profiles can also be seen in figure 23.

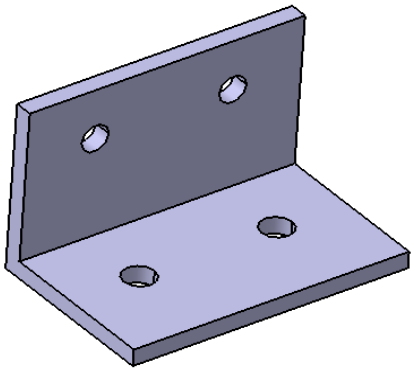


Figure 23: L-profile

Both the platform and the L profiles were manufactured at the Hochschule "Shop Floor" at the Hochschule Osnabrück. The steel platform was painted anthracite to maintain the harmony of the laboratory. Before painting, it was necessary to give a coating of primer to the steel in order to prevent corrosion. Thus, a first coating of primer was applied und left to dry. After the first coating of primer was totally dried, the first layer of anthracite painting was applied and left to dry. After a whole weekend, the second and last layer of painting was applied and finally left to dry. The L-profiles were also painted but without the application of primer as it was not considered necessary.

In figure 24, the platform while the first layer of primer was being applied is shown. The red colour is the steel itself whereas the grey colour is the applied coat of primer. As it can also be observed, the floor of the laboratory was covered by cardboard so that the floor would not get painted accidentally. The process of painting could not be performed in the special room for painting because of the size of the platform. The platform was wider than its door and it was impossible to get it into the room by using the hand pallet truck.

Figure 24: platform and first layer of primer

3.1.2. NEW STRUCTURE FOR THE KAWASAKI

In second place, and as it has been introduced before, the structure where the Kawasaki was installed was constructed of aluminium profiles and it was united to the big profiles structure of the cell. It was a real problem so a new structure that would be united directly to the new platform was needed. There was an old structure in the laboratory of an old robot that could be used but first modified. It was measured and decided that it could be used. This structured had four pins that united it to the old platform but, as the main point was to achieve stability, all these four pins were removed in order to screw it directly to the platform. Moreover, the tube structure had to be cut because it was too big and only one half of it was enough. And so was the platform that was on top of this structure where the Kawasaki was going to be located. Thus, it was sent to the Shop Floor of the Hochschule Osnabrück.

In order to locate the Kawasaki on the plate on top of the modified tube structure, and regarding the possibility of working on both the assembly line and the work table, it was design in order to change its position. Thus, six holes were drilled to the plate and when it works on both sides (work station and work table at the same time), the robot would be located on the left side of the plate in figure 25 (or top four holes in figure 26) and if in the future is decided to locate it away from the assembly line, it will be located on the right side which is centred with respect to the work table. The plan for this plate can be found in annex 2 as plan 3.6.

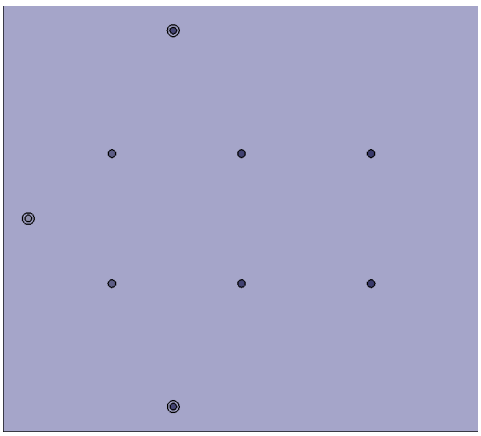


Figure 25: plate of the robot

In figure 26, a top view of the whole cell of the Kawasaki is shown. It is the best way to explain why the existence of the six holes. If the robot is located at the centre of the table, the work station would stay too far from it. Furthermore, there would be some incongruity points with the six joints when the robot would try to reach the farthest point of the tray. That is why the two-position possibility was decided. Thus, when the robot would stay next to the assembly line and would work on the assembly station, it would be located on the nearest four holes as it can be seen in figure 27. Otherwise, the robot would be located on the centred four holes in order to be centred looking at the table.

WOKING STATION

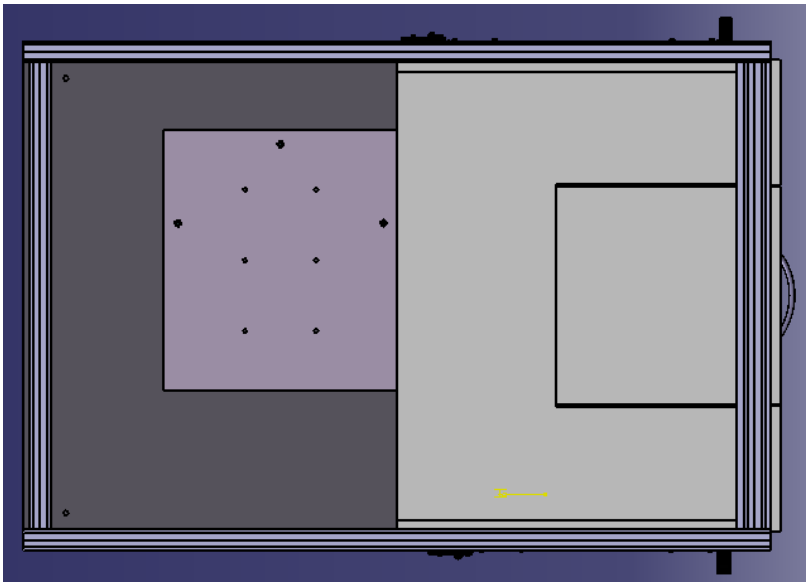


Figure 26: top view of the cell

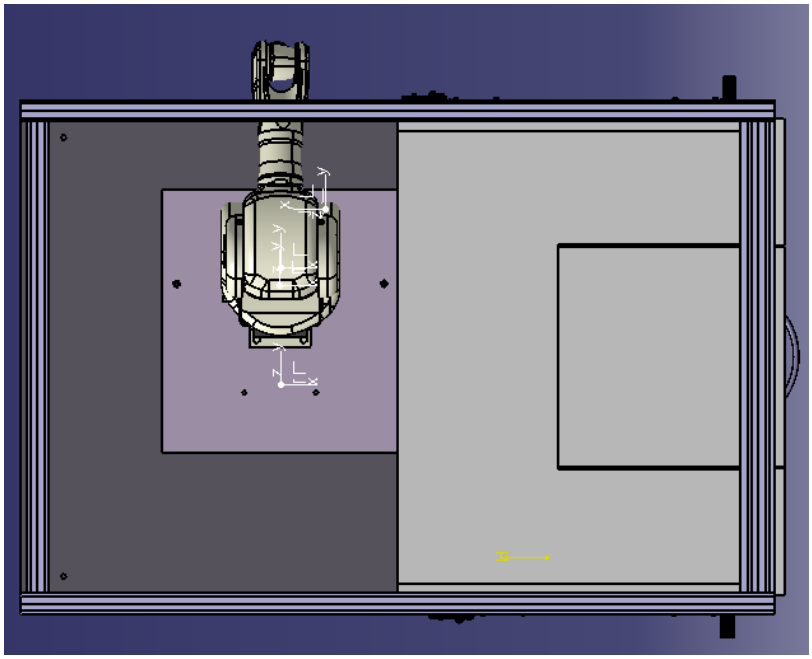


Figure 27: robot working on the work station

To end, figure 28 will show a 3D design of the whole platform including the structure for the Kawasaki and the robot itself. After it, in figure 29, a 3D design the complete cell of the Kawasaki is shown. The robot would be looking straight to the work station in this figure.

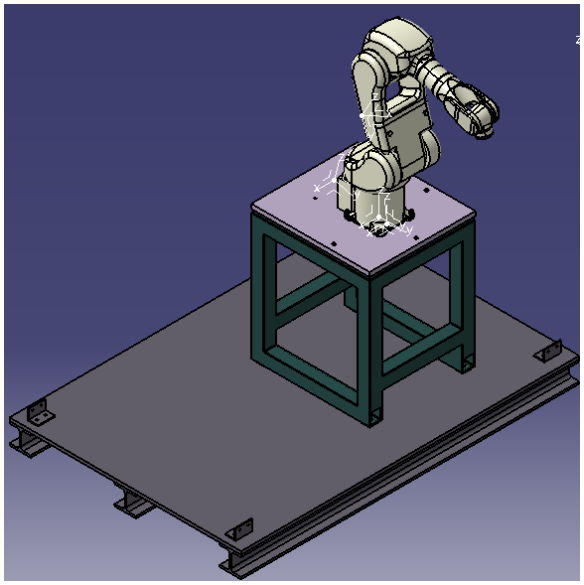


Figure 28: new platform and structure for the robot

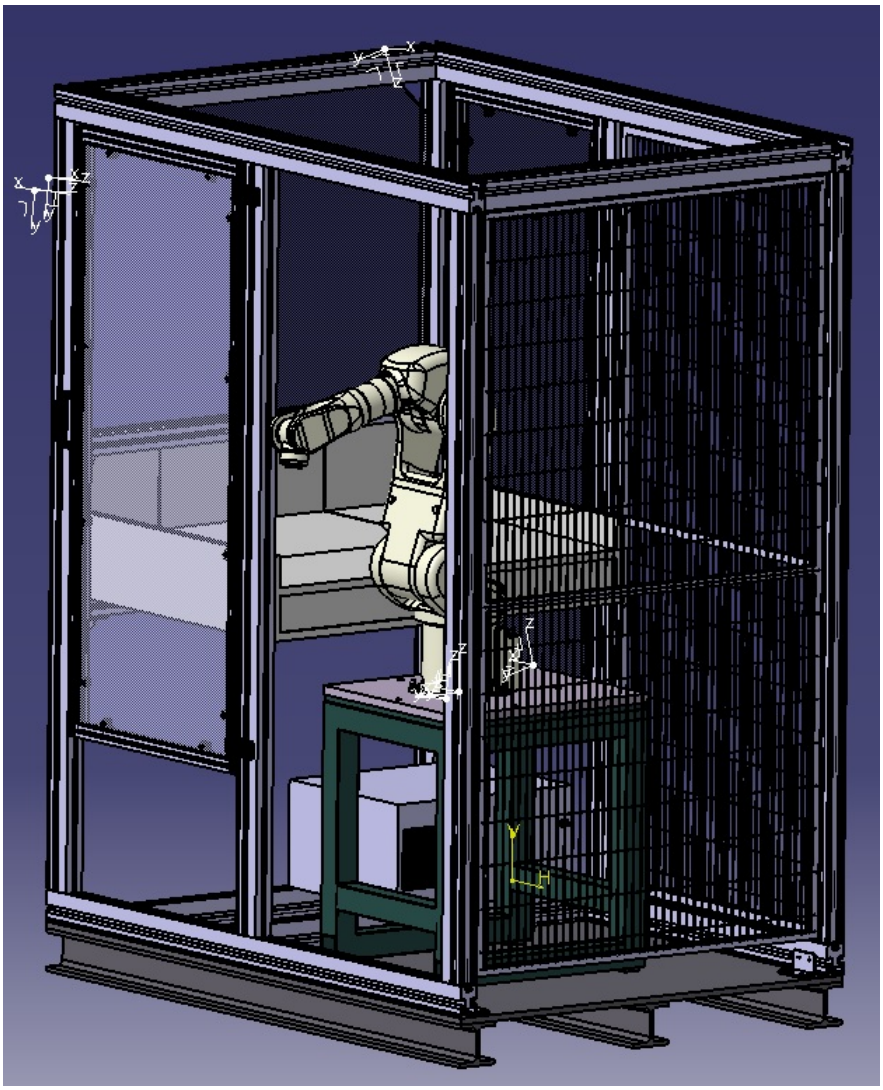


Figure 29: new design of the cell

3.2. BUILDING THE IMPROVEMENTS OF THE NEW CELL

Once all the new components needed were manufactured and the platform was painted, the next step was to build everything together.

The first step was to disconnect all the wires and cables from the Kawasaki because it had to be placed on top of the new structure. The Kawasaki was located on the left side since now it will be working on top of the work station. It was screwed with four bolts of diameter 6 mm to the top plate of the structure. After it, the structure itself was screwed to the platform using another four bolts but in this case its diameter was 12 mm. As the structure was already installed and due to the rectangular form of the cell, the front grill had to be removed in order to locate the whole cell on the platform. For this step, the whole cell was lifted using the hand pallet truck in the laboratory. The old union floor-cell was removed in order to rest the whole cell on top of the platform. It was no easy to do as the movement of it from the hand pallet truck to the platform had to be done by hand and the cell was heavy itself. After this step, some painting was damage and it had to be repainted.

When the cell was on the platform, it had to be left like this because the L-profiles were not jet manufactured. After the L-profiles were received, they were painted and located one in each of the four corners of the structure and screw them to both the profile of the cell and the platform through the holes drilled before in the double-T profiles and the big plate.

Once all the structure itself was installed, all the wires and cables were reconnected again to the Kawasaki and these were organized to get a better view of the Kawasaki's whole cell. Also a new profile was installed on the base of the cell just at the back of the Kawasaki in order to gain a little of stability and to keep the two largest profiles together. This new profile is located just below the robot.

Some of the components like the pressure valve and some other sensors were screwed to the old structure of the Kawasaki. With the new structure, it was impossible to screw them to it so a small profile was located in order to hold the pressure valve and the other sensors were screwed to one of the big vertical profiles of the cell. It can be seen in figure 30.

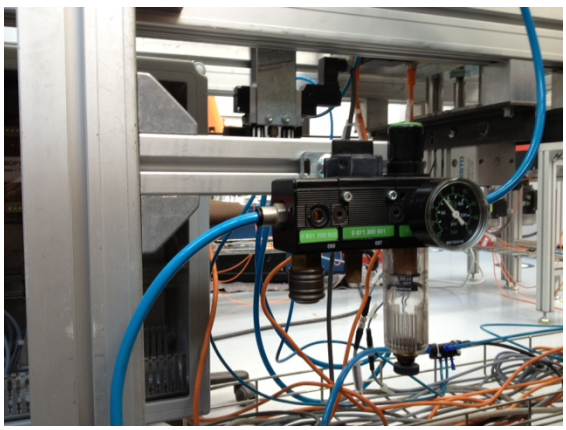


Figure 30: new location of the pressure valve

Finally, some real images of the cell will be shown.

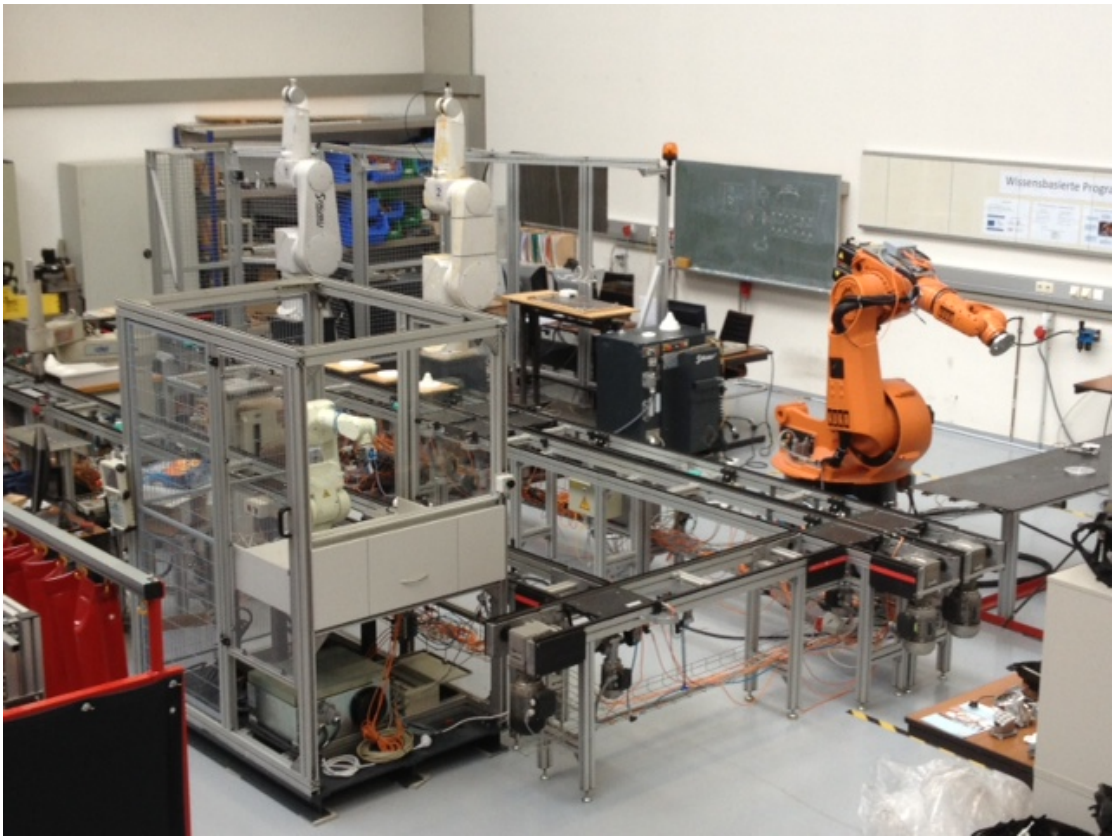


Figure 31: assembly line

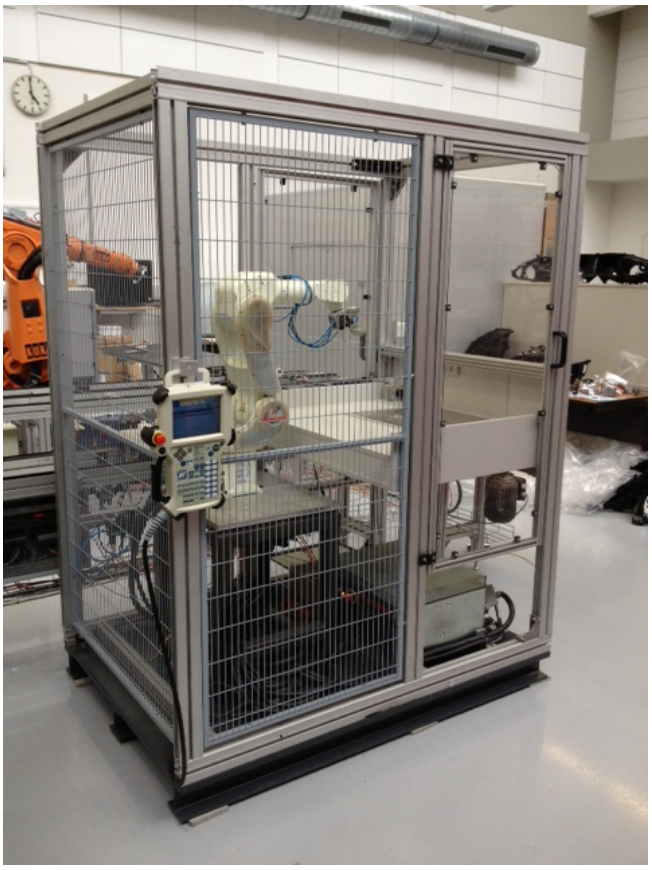


Figure 32: work cell

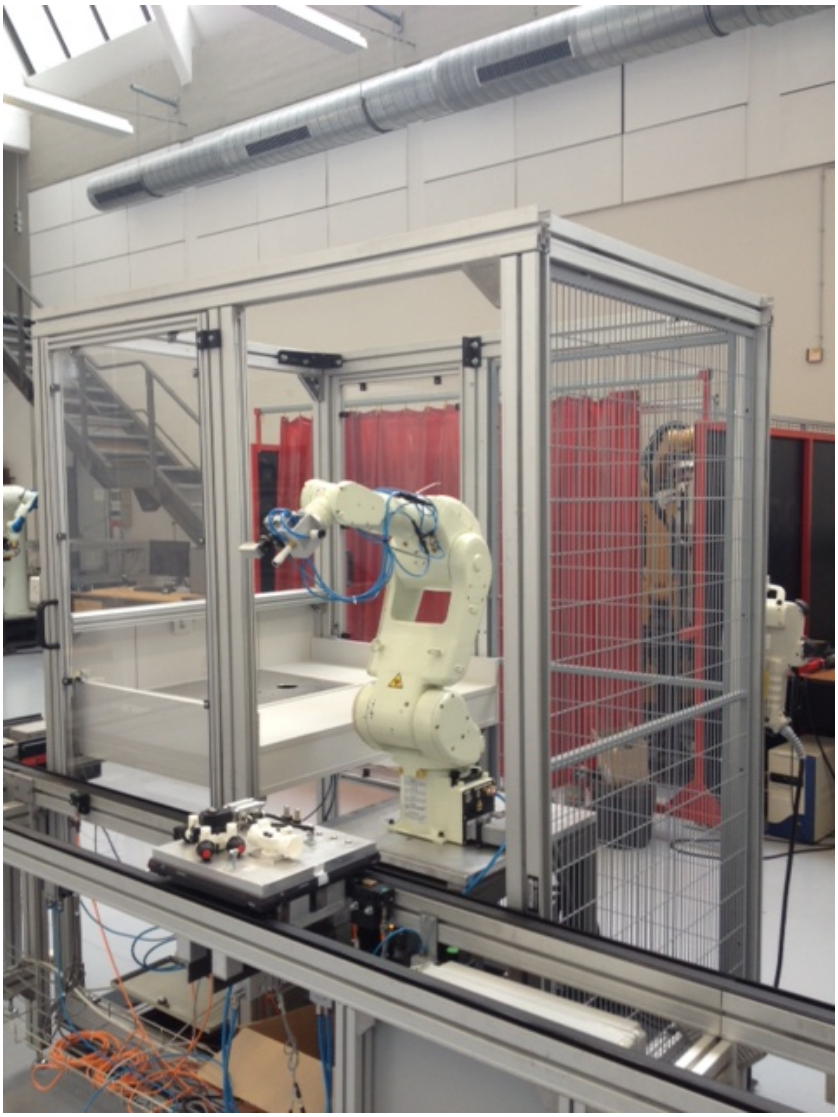


Figure 33: cell 3

In figure 31, a top image of the assembly line can be seen. It can also be seen the other robots located by the same assembly line. In figure 32, an image of the robot work cell is shown. It can be seen the new platform and the new structure of the robot. Furthermore, the robot is working on the work table where as in figure 33 the cell can be seen from another point of view. In this figure, the robot is working on the work station. This work station had some problems that are going to be explained next.

3.3. FIXING OF THE WORK STATION

The work station located in front of the Kawasaki was not working. At the beginning, the pneumatic system was not installed. There was one pneumatic system next to the robot and it was installed in order to fix it. Once installed, it did not rise when it was required from the PLC so there was one problem out of the next two: whether the valve or another component of the pneumatic system was broken or the signal from the PLC was not right or disconnected. To clarify this error, the pneumatic system from another work station was taken to the Kawasaki's work station. It was installed and this time it worked perfectly when it was required from the PLC. So it was concluded that the problem was from the valve or the pneumatic system. In next figure 34, the work station can be seen.

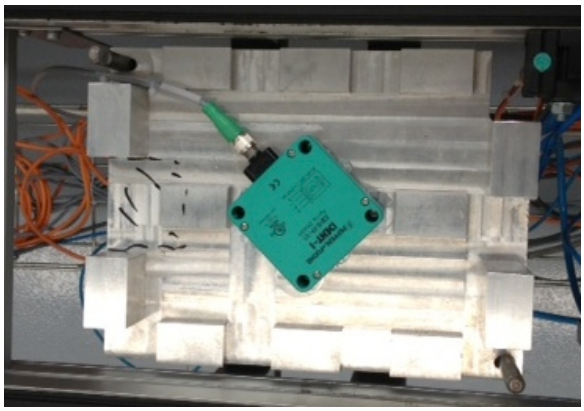


Figure 34: work station

A new pneumatic system had to be bought for the Kawasaki's work station and it had to be the same model installed at the other work stations. Everything needed was from Pepperl + Fuchs, GmbH:

- One magnetic valve with reference **MEH-5/2-1/8-B.**
- One Electrical accessory with reference **MSSD-E.**
- Two quick star glands with reference **QS-1/8-8.**
- One quick star gland with reference **QS-1/8-4.**

After the pneumatic system was correctly installed, the next problem was that the pins did not fit into the holes of the tray, which was on the work station. So when the station rose, as the pins did not fit into the holes, the tray stayed not straight doing it impossible to work on it. At first, it was thought that the pins had a bigger diameter than the holes but after comparing those with the pins at other station, it was realized that that was not the problem. After it, the whole work station was compared with the nearest one and it was perceived that the exit stopper was located too far away from the station itself and that the pins were screwed too close one against the other. The solution was to screw these with the correct distance between these and to move the exit stopper until the pins fit into the tray's holes when it rose.

In the next figure 35, the pneumatic valve can be seen. The small blue tube at the top is the tube through which the air comes from the pressurized air system. The two big blue tubes, are the tubes for both two positions, when the work station is up or down. Finally, the orange one is the signal which goes to the control program of the PLC and which controls the station.

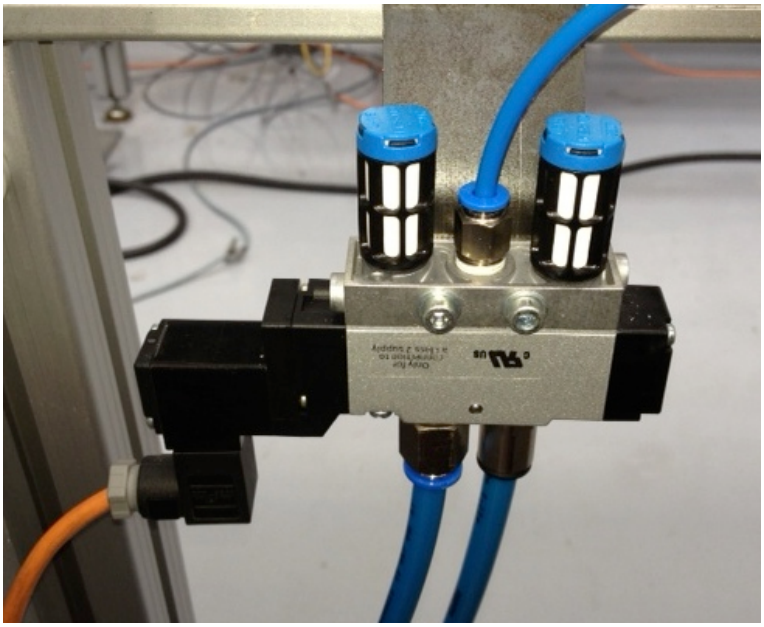


Figure 35: pneumatic valve

4. DESIGN OF A NEW ASSEMBLY OPERATION

4.1. DECISION OF THE NEW ASSEMBLY OPERATION

The final achievement of this master's thesis is the implementation between the robot and the PLC. For this reason, there is not a better way to conclude it than performing the assembly of one product by programming the robot in order to be fully controlled from the PLC while it sends all the information the PLC would require. As it will be explained later, the robot will send information about the step it is working on, number of steps, etcetera.

Everything related with the software will be explained in the next chapter 5. In this chapter everything related with the assembly product and the assembly process is explained.

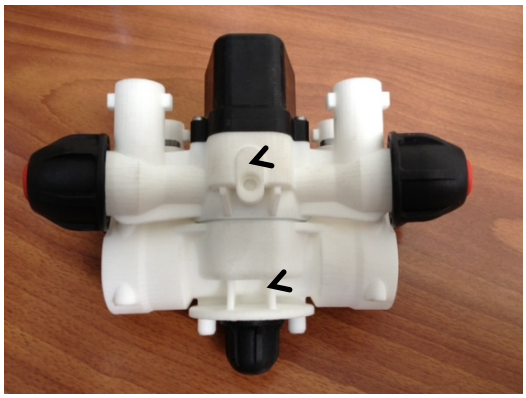
After all the robot work structure including the new platform and the base for the robot were performed, the next step was to decide the product to be assembled. There was one student from the Hochschule Osnabrück writing his "Bachelor Arbeit" in the company Amazone located in Hasbergen-Gaste. Hasbergen-Gaste is a small town in the vicinity of Osnabrück.

In its website this could be read about Amazone: "High-tech for a booming industry - that is our objective. We from Amazone develop and produce innovative agricultural technology enabling agriculture to meet the future continuously increasing demand on food and alternative energies" (12).

It was decided to contact this student to ask if they had a suitable product to assemble together. A box containing one product was received and discussed to see whether it was suitable or not. This product was a sprayer for an agricultural vehicle. It was not a final product but a rapid prototyping prototype. It consisted in one cavity where the fluid (water, fertilizer or what would needed to be sprayed onto the field) entered from one side and with an electronic device. This fluid Is sprayed on the field through one out of the four existing exits. Each of the four exits had the target of changing the shape of the fluid at the outlet.

This product can be seen in figure 36. It is composed of two big parts screwed together with two bolts. Inside of the two parts, a small cavity can be found where four springs are located. On top of each spring there is a small metal ball. There is also a small metal cam that can rotate around one axis. This cam moves one ball at a time compressing the spring and allowing the fluid to exit through its hole.

To make it easier to understand, from now on each of the two big parts of the assembly product will be given one number. The part on top (the one with the four plugs) will be named part 1 and the part on bottom will be named part 2 as it is shown in figure 36.



- **PART 1**
- **PART 2**

Figure 36: assembly product

There were two suitable possibilities for the assembly of this product. The first one was to place part 1 at the bottom. It would be already located on the tray. After it, the four balls would be picked up and easily placed one by one onto its place. Then, the part 2 would be located on top of the four balls and the part 1. This part 2 would carry the four springs with it. The main disadvantage of this possibility was that there was no chance to screw the bolts to get one final product, as the bolts would be located on the underside of the whole assembly product. A step diagram of this process can be seen in the following

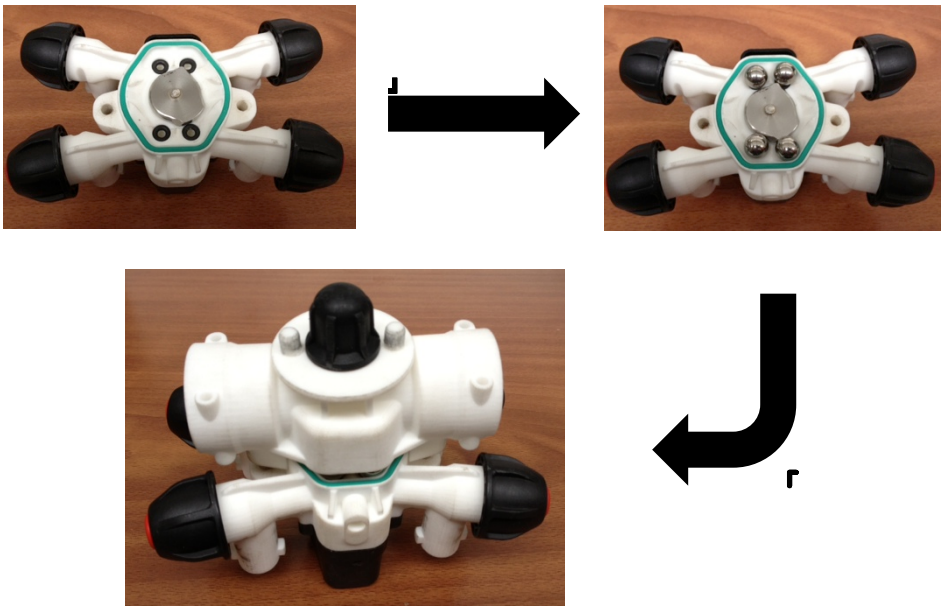


Figure 37: assembly operation, option 1

On the other hand, the second possibility allowed obtaining a final assembled product as the two main parts would be screwed one to the other. For this assembly operation, the part 2 would be at the bottom. This part 2 will already carry the springs with the aim of making it easier. Otherwise, the assembly operation might be too tricky. The second step would be the pick and place of the four balls on top of each spring one by one. Then, the part 1 would be

located on top of the balls and the part 2. After it, each of the two bolts would be located into its wholes and screwed. The final product would be the whole assembled product. To understand it easier, the next couple images show the whole process of assembly (see figure 38).

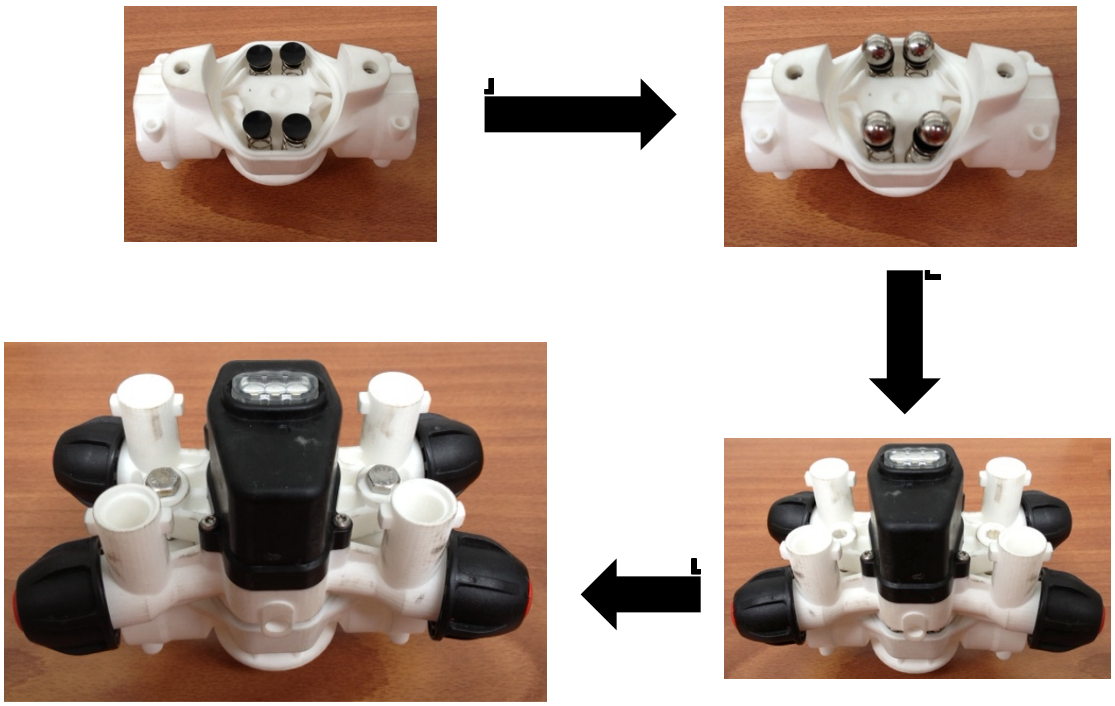


Figure 38: assembly operation, option 2

Due to the advantage of obtaining the whole assembly product together and even though it would be more complicated, it was decided that the second possibility would be performed.

4.2. DESIGN OF THE NEW ASSEMBLY TRAY

All the components for the assembly operation come to the work station on top of a tray that is moved along the assembly line by a conveyor belt. It simulates one pre-process that would take part in other previous steps. All these parts must be located in a determined way so that the robot will pick these from the same place each time and work with them as needed. For this reason, a specific tray must be manufactured. For the manufacture of this tray there were two possibilities.

The first one was to manufacture it out of wood. It would be cheaper but the final product would not be as good as if it was of aluminium. That is why it was decided to make it of aluminium so it would be more practical and with better precision. In figure 39, the first design of it using CATIA 3D is shown. The dimensions of the plastic tray with the RFID system are 310 mm x 310 mm so the dimensions of the aluminium plate should be the same. The thickness is 20 mm but there is a small plate of 5 mm on top of it to locate on top of it one of the big parts. For further technical information see plan 4.1 in annex 2.

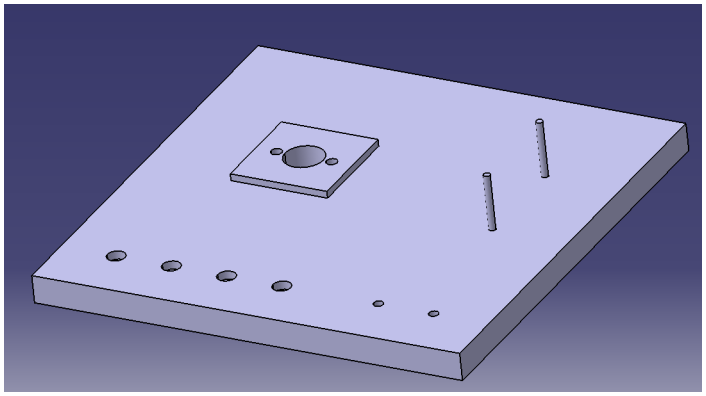


Figure 39: first design of the tray

Each part of the assembly product is located in one place of the plate. For example, the two big parts will have its own location at the back side of the tray whereas each ball has a hole where it is introduced waiting for being picked up by one tool. The same happens with the bolts; there are two holes to fit them inside at the front side.

Some improvements or redesigns had to be performed after the first design due to some problems found when the assembly operation was being programmed. These changes will be explained once the whole assembly process is explained explaining what was needed for each step and the changes or improvements that had to be done for each step.

So, in the end, the tray will arrive at the work station, the PLC will send the order for it to rise and then the PLC will allow the robot to start its work. Once the robot has finished it or an error has occurred, the robot will communicate it to the PLC and the work station will go down allowing the tray to continue its way. The assembled product will be on the tray. All the communication and the interaction between the robot and the PLC will be explained in chapter 5.

4.3. DESIGN OF THE NEW TOOLS AND THE NEW ADAPTER OF TOOLS

The use of only one tool severely limited the decision of the assembly operation. Due to the different number of parts and especially to its huge difference in shape and size, more than one tool were needed. The parts, which must be picked and placed into its place, are the four metal balls, part 1 on top of part 2, and the two bolts (see figure 38). The only possibility for this process was the use of one tool for each of the different parts. Thus, one tool had to be designed for each part.

4.3.1. TOOL FOR THE BALLS

In order of use in the assembly operation, and as it will be explained more in detail, the first task would be to locate four balls on top of respective springs (see figure 38 upper right). As the use of a gripper could be too tricky, the determined solution was to use a vacuum tool. It would be much easier to pick the ball up with the vacuum force. For this reason a Venturi valve has been used. This valve works regarding the Venturi effect. Under this valve, there is a sucker. So, this tool will have to be moved on top of the ball, then the vacuum force will start and the sucker will suck the ball. It will be moved to the final position and it will place the ball there by ending the vacuum force.

“The Venturi effect is a jet effect; as with a funnel the velocity of the fluid increases as the cross sectional area decreases, with the static pressure correspondingly decreasing. According to the laws governing fluid dynamics, a fluid's velocity must increase as it passes through a constriction to satisfy the principle of continuity, while its pressure must decrease to satisfy the principle of conservation of mechanical energy. Thus, any gain in kinetic energy a fluid may accrue due to its increased velocity through a constriction is negated by a drop in pressure. An equation for the drop in pressure due to the Venturi effect may be derived from a combination of Bernoulli's principle and the continuity equation” (13). Taking advantage of this drop of pressure, the ball would be sucked and placed into its final position. The fluid used would be pressurized air using the system in the laboratory.

A real image of this new tool can be seen in the next figure 40:

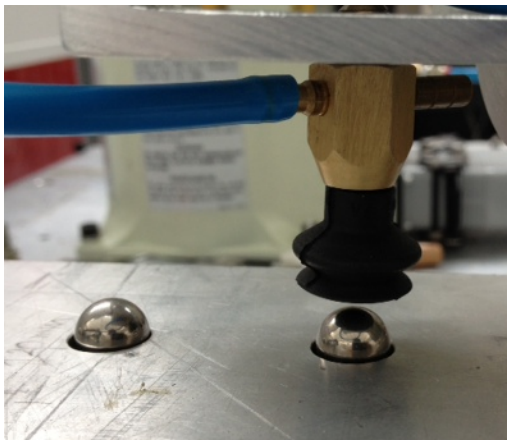


Figure 40: tool for the balls, balls in old stock

4.3.2. TOOL FOR PART 1

In second place, to pick part 1 and to place it on top of part 2, only a gripper would be needed. There were some grippers already in the laboratory. Thus, there was no need to buy a whole new tool but the redesign of its fingers. The only flat surfaces, that were located one in front of the other enabling its collection in an easy way, were located at a distance of 60 mm. In the next figure 41, the gripper with the new fingers can be seen. This tool will locate on top of the part 1 with its flat fingers parallel to the flat surfaces of it. It will close the fingers holding the whole part and moving it to its final position. Once there, it will open its fingers leaving it there. For further technical information see plan 4.2 in annex 2.

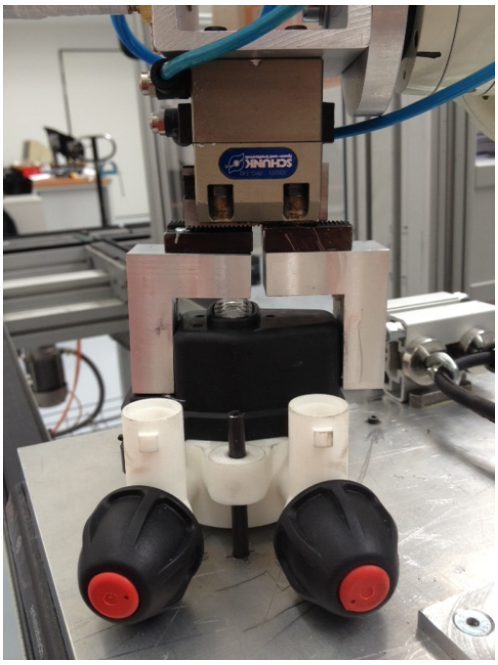


Figure 41: tool for part 1

4.3.3. TOOL FOR THE BOLTS

Last but not least, the bolts must be located into their wholes and screwed as much as possible in order to get the final assembled product. At the beginning, the use of a magnetic tool was taken into consideration. With this tool there was the need of two tool adapters in order to pick up both the bolt and the screwdriver always in the same position. Afterwards, the design of a new tool that already incorporated the screwdriver took more importance than the first idea. Not only would this new tool have a vacuum system in order to pick up the bolt and place it into its hole, but also the head would have a hexagonal section in order to work as a screwdriver and screw these.

Using one hexagonal head already existing and another Venturi's valve, the only part of it to be design was the body that would unite the both sides. The solution was a hollow cylinder that was screwed to the valve and welded to the hexagonal tool. This idea was taken to the Hochschule "Shop Floor" and with a small variation it was manufactured. A picture of the

final product can be seen in figure 42. The plan taken to the Shop Floor can be found in annex 2 as plan 4.3.

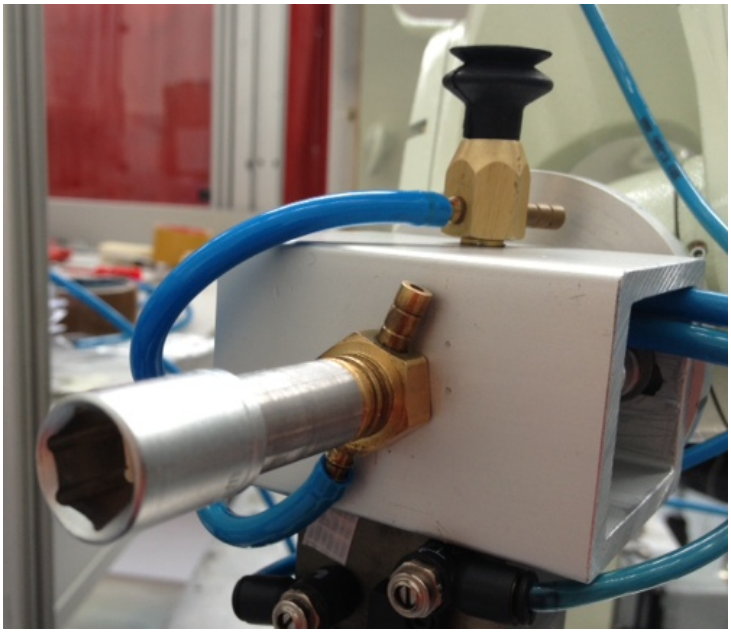


Figure 42: tool for the bolts

4.3.4. TOOL ADAPTER

At this point, the three needed tools were designed but there was not the possibility to for the Kawasaki to change the tool between each step of the assembly operation. Thus, something was needed to use all these three tools at once. The solution was the design of one tool adapter. It would be screwed to the Kawasaki and the three tools would be located on it.

The only requirement was that the screwdriver should be coaxial with the joint number six of the robot in order to turn all the possible 720 degrees for the purpose of screwing the two bolts. The design of this tool adapter was done given that no tool would hit any part of the assembly operation no matter it was being used or not. For this reason, a rectangular tube was the best solution for the tool adapter. The original design was a tube of 70 mm x 70 mm but the only material available at the Hochschule “Shop Floor” was a tube of 60 mm x 45 mm so a redesign of it had to be done. The final design of this adapter can be seen in plan 4.4 in annex 2.

Once all the tools and the adapter have been described, there is not a better way to imagine it than seeing a real image of it already with the three tools. It can be seen in figure 43.

Figure 43: tools adapter

4.4. IMPROVEMENTS OR CHANGES OF THE ASSEMBLY OPERATION STEP BY STEP

The assembly process must be divided into main steps. Each step should work independently so that the PLC can instruct the robot the first step that has to be performed. For example, if the PLC requires starting in the third step, the two first steps do not have to be performed. This process could be easily divided into five or nine steps depending on whether the process of placing the balls on part 2 is divided into four steps or left as a single one. As the balls could not stay in place if part 2 is moving on top of the conveyor belt, it makes no sense to divide this process into four different ones. Thus, the final process will be divided into 5 main steps listed on the following list:

- Step 1: the four balls are picked and placed on top of the springs.
- Step 2: part 1 is picked and placed on top of part 2.
- Step 3: the elastic is placed on top of part 1 and part 2 in order to put the two big parts together.
- Step 4: the first bolt is picked and screwed to the nut inside part 2.
- Step 5: the second bolt is picked and screwed to the other nut inside part 2.

Each step will be described more in detail showing the improvements or changes needed for a correct assembly. These steps will be described in order beginning with the first step and concluding with the last one.

4.4.1. STEP 1: PICK AND PLACE OF THE FOUR BALLS

As it has been said before, each ball comes placed in one of the four holes on the tray. When the tray moved along the assembly line, as each of the work stations has two stoppers (one at the entrance and a second one at the exit), the tray was stopped there violently and sometimes the balls moved out of their holes. Thus, there were two options: to drill the holes deeper or to glue a washer with a same diameter on top of the tray. The solution decided was the second as it was easier to perform (see figure 44). In this step, the tool must be located on top of it, it must descend and place just on top of the ball. Next figure 44 shows the tool descending to the ball (it must still descend some millimetres).

Then, the valve must be closed so that the vacuum force starts and the sucker sucks the ball. Once the ball is sucked, the tool will move above the corresponding spring. After, it will descend until the ball almost touches the rubber part on top of the spring. Once there, the valve must be opened stopping the vacuum force and it has to wait there for almost one second until the balls falls from the sucker.

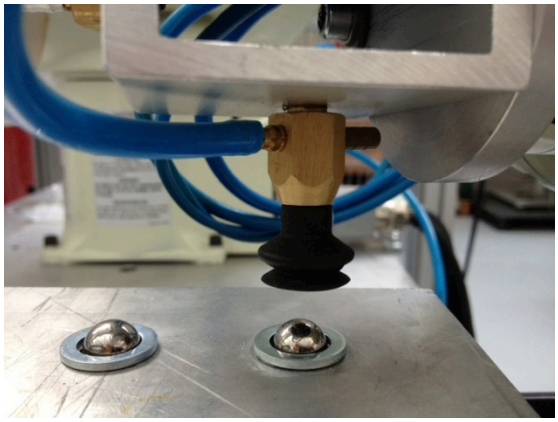


Figure 44: tool descending above the ball

As the spring might be not vertically located, the tool will descend some millimetres compressing the string and locating it vertical. After it, the tool will ascend but it will stop where the spring would be again not compressed and will wait there almost another second until the ball gets stabilized and does not move. When it would stop moving, it will never fall from the spring. The tool will ascend and will start with the next ball. The transport of the ball can be done at maximum speed but when the ball is picked or left, the speed must be much lower so that everything would work properly. In next figure 45, the tool is leaving the third ball on top of its spring.

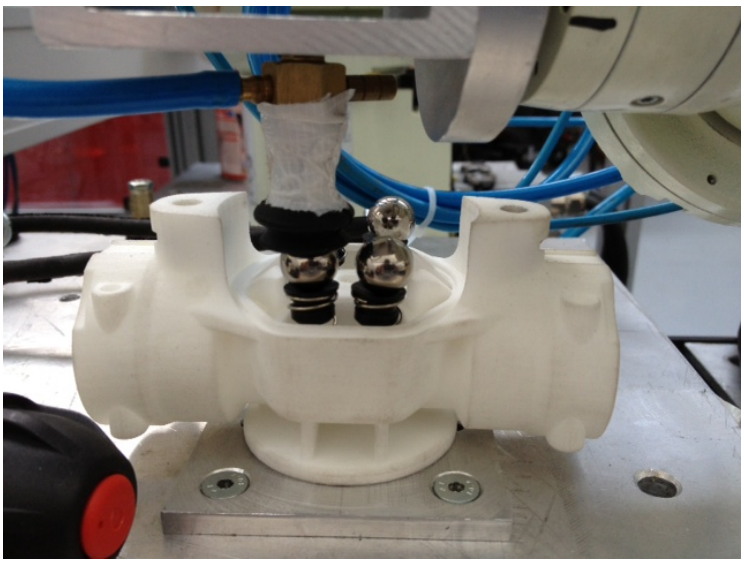


Figure 45: tool leaving the ball

4.4.2. STEP 2: PICK AND PLACE OF PART 1

The second main step is the pick and place of part 1. After the four balls have been placed, the joints of the robot will move in order to use the gripper. With the new fingers placed on this gripper, it picked part 1 and placed it on top of part 2. Part 1 is located on top of the tray by using two pins as it can be seen in the next figure 46:

Figure 46: part 1 on the tray

The gripper will be located above part 1, it will descend slowly on top of part 1 and then the valve will be closed closing the fingers of the gripper. Then, the tool will ascend leaving below the pins and will move on top of part 2. Then it will descend slowly until the two parts are in contact. The valve will be opened and the gripper will ascend leaving part 1 there. The first main problem comes now when the gripper leaves part 1 on top of part 2. As inside of these there are four springs, part 1 and part 2 do not stay together. Furthermore, this is a problem when the bolts must be located into the holes. As part 1 moves upwards, the two holes might not stay coaxial and the bolt cannot go through these easily or in case these stay coaxial, the two parts will stay too far one from the other doing impossible to screw these together. Thus, something must put these two parts together when the gripper leaves part 1 on top of part 2. The solution for this is explained next. In the next figure 47, the two big parts can be seen once the gripper has left part 1 on top of part 2.

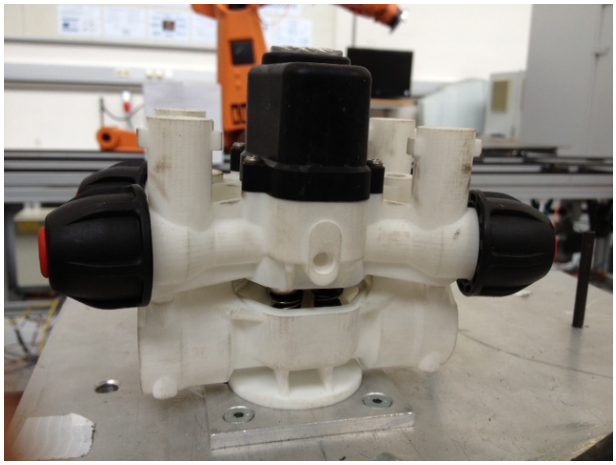


Figure 47: part 1 on top of part 2

4.4.3. STEP 3: ELASTIC BAND

After some ideas such as the manufacture of a pusher next to the screwing tool that would push the two big parts one to the other, the final idea was the use of an elastic band. This elastic band would be placed on top of both parts and will force the two parts downward putting these together. The next step was to design how to locate it on the parts and how to

fix this elastic band to the tray. Logically, the use of some rings came at first. One of these rings would be located directly on the tray while the other should be moved above the two parts and then should be fixed somehow to the tray.

This movement had to be done by using one of the three existing tools. It is difficult to imagine doing this with a vacuum tool since it could not counteract the force of the elastic band. Therefore, the only possibility is to use the gripper. As it was designed to pick part 1, it could only pick up parts with a length of about 60 mm. So, the other ring should be placed on something of this length. The final idea was to take a square profile of 60 mm and to screw the ring to it. After some problems using only one ring screwed to the profile, it was realized that it would be easier to work when three rings would be used: one screwed directly to the tray and the other two rings screwed to the profile so that it would be easier to place the two parts of the elastic band around part 1.

In order to screw the ring directly to the tray, a new hole had to be done. It was drawn on the tray and the tray was taken to the Hochschule “Shop Floor”.

The final design of the tool is shown in the next figure 48.

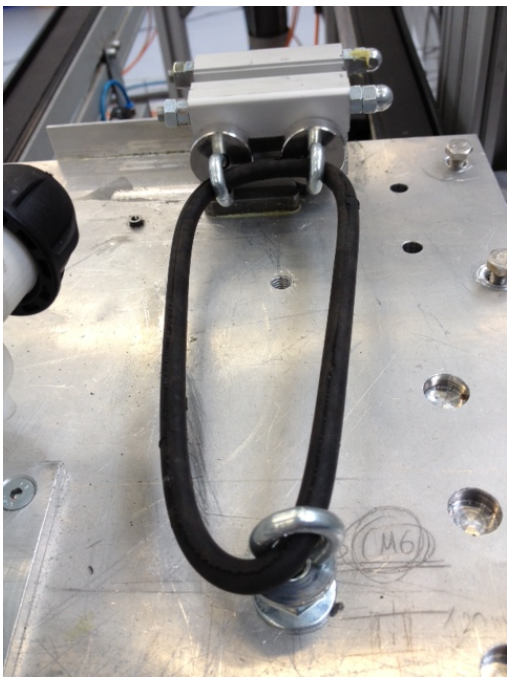


Figure 48: new tool for the elastic band

After it, the next design should be the initial and final location of this tool. It had to have its own space. The initial position of part 1 on the tray had to be changed in order to make room for this tool. Nevertheless, only one of the pins was changed and the whole part was turned around the axis of this pin. This displacement allowed getting some space for the new tool.

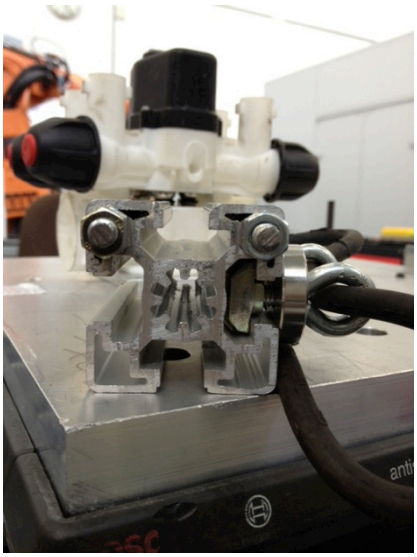


Figure 49: profile section

Taking advantage of the section of the profile (see figure 49), a design using one bolt, which would fit this section, was thought. The final result is shown in figure 50.

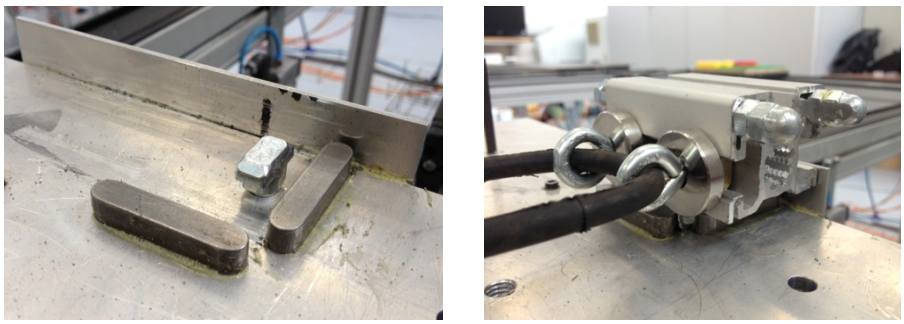


Figure 50: initial location of the elastic band

The bolt will get inside the profile whereas the plate and the two steel parts will make sure that it will always be at the same position at the beginning. These elements will also guide the profile in its way out of the bolt going to the left.

Once the initial position has been defined, the gripper will be placed over the profile, it will descend and it will pick the profile with its fingers. It will go through the programmed path so that the elastic band will leave the initial position, will be correctly placed on top of the two parts, and will take the profile to its final position pushing the two parts together.

Using a bolt for the design of the final position was also thought. It would be screwed directly to the aluminium plate and the only point left was the height of it. It was tried with some heights until it worked the best. The profile will be moved through the path and, at the end, it will be located in a way such the bolt will be located inside this profile. As the bolt had to be placed in the middle of the profile in order to stay steady and with the problem that the two parts moved when the whole profile was moved in order to enter the bolt from one of the sides, the profile had to be redesign. The solution was to cut out part of the profile to make the whole process easier. The final design can be seen in the next figure 51.

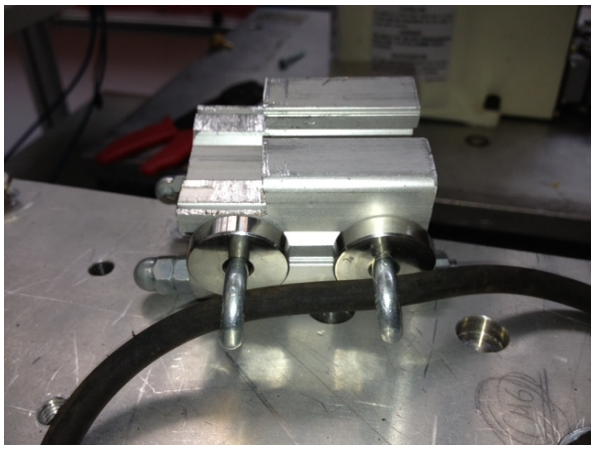


Figure 51: redesign of the profile

Finally, the next two images in figure 52 show the initial and final position of the elastic band:

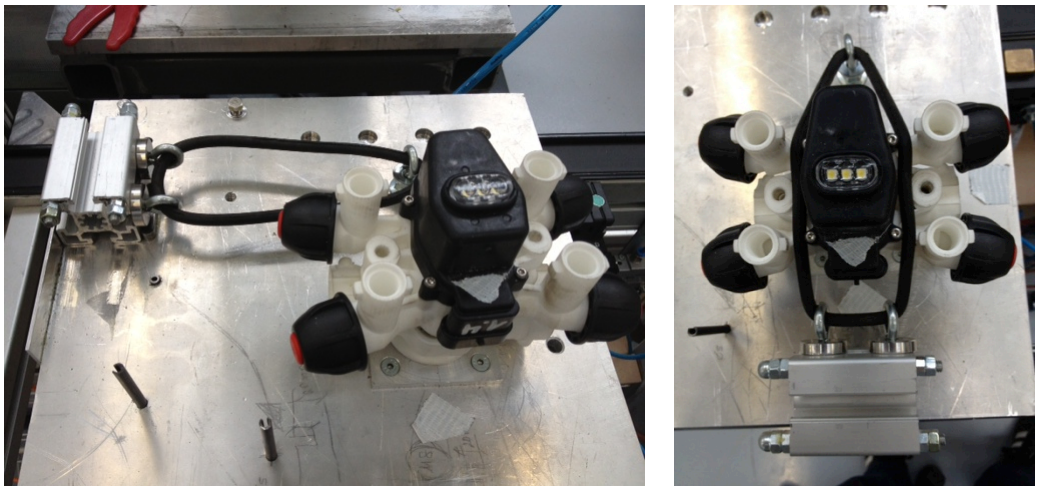


Figure 52: initial and final position of the elastic band

As it might not be clear enough the procedure of the final position of the tool for the elastic band, the next image will clear this design (figure 53).

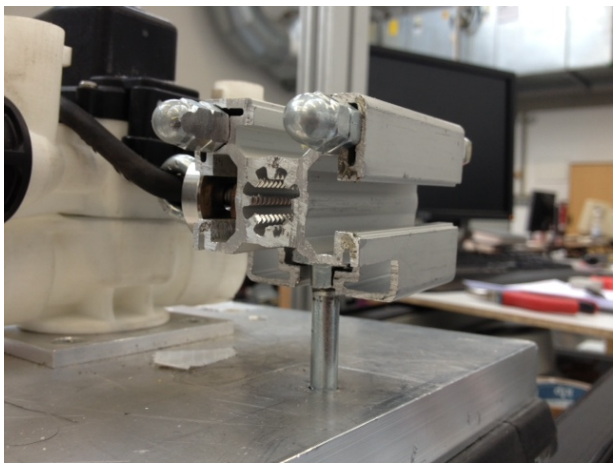


Figure 53: final position

There was a last problem with this design. As the elastic band was stretched, its force was greater than the force of the gripper and the profile moved inside it doing impossible to locate the profile around the bolt as every time it moved in a different way. Because of this, something had to fix the fingers to the profile preventing relative movement between these. The solution was to screw two bolts along the profile doing the distance between them the thickness of the fingers (25 mm). With this solution, no matter how great the elastic force is, the profile remains always at the same place. This can be seen in the next figure 54.

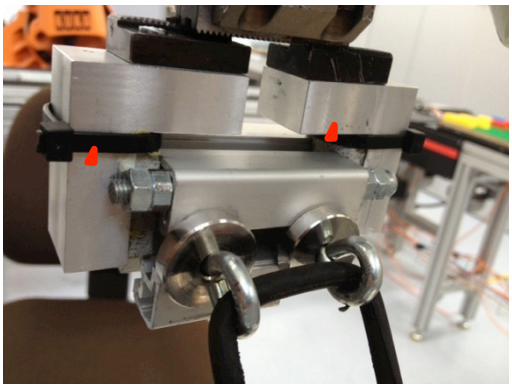


Figure 54: bolts for the fingers

As the elastic force was great, the whole product moved out of its position, so a little push down by one of the fingers had to be performed after the elastic band was left to its final position.

Finally, the two needed holes, both for the initial and the final position bolts, had to be drawn and the tray was sent to the Hochschule "Shop Floor" at the same time as the hole for the ring. These holes were of diameter 8 mm whereas the one for the ring was of diameter 6 mm.

4.4.4. STEPS 4 and 5: PLACING THE BOLTS

Finally, the two holes for the bolts did not work as expected. Therefore, a new solution for the bolts had to be found. The tool has to descend and pick the bolt by using a vacuum force. Thus, the head of the bolt has to be in contact with the internal surface of the tool so that the vacuum force can be exerted on top of the bolt and it can be picked.

Because of this, one specific support had to be designed for the initial position of the bolt. This structure has to be designed so that the tool can descend and touch the bolt. On the other hand, it should have some freedom in order to move a little to its sides so that the bolt can descend properly.

The first idea was the manufacture of a support that would keep the bolt fixed always in the same position. This support was too rigid and there were many problems and it did not work properly. Thus, another solution had to be thought. The final solution can be seen in the next figure 55. This final idea is composed by one base and one body. The base is a rubber ring. This rubber ring gives the whole support some freedom as it can be compressed or

stretched when the tool descend in order to adapt it to the tool. The body is one nut. This nut has an external diameter of 9 mm and the bolt has a hexagonal head of 10 mm of side. The head is bigger than the nut so when the tool descends, it only touches the bolt and it allows the tool to pick the bolt without further problems. This nut had to be drilled in order to remove the thread as it hindered the exit of the bolt. An image of the tool picking the bolt can also be seen in figure 56.

Figure 55: initial position of the bolts

Figure 56: tool picking up one bolt

The tool will be located just on top of the bolt. Then it will descend turning around its axis so that the bolt gets into its position inside the tool. Once there, the valve will be closed and the bolt will be sucked. After it, the tool will place it above the hole of part 1, the valve will be opened and the bolt will descend into the hole. As it might not descend totally, the tool will ascend leaving the bolt there, it will move some millimetres outside of the axis and will descend pushing the bolt with its edge. When the bolt is correctly placed, the tool will move and will locate around it. Then, it will descend turning again so that the bolt gets into its place once again and it will revolve about 500 degrees so that the bolt will be screwed to the nut and the two big parts are joined together. After it, the tool will ascend and will repeat this step with the second bolt.

4.4.5. FINAL DISTRIBUTION OF THE TRAY

After all the required changes and improvements of the tray and new tools, the final distribution of the parts on top of it is very different from the initial 3D design using CATIA (figure 39). Only the holes for the balls (adding the washers) and the position of part 2 have remained in the same place and in the same way. In the next figure 57 the final distribution of the parts on the tray are shown. Furthermore, in figure 58 the final distribution of the parts after the assembly operation are shown.

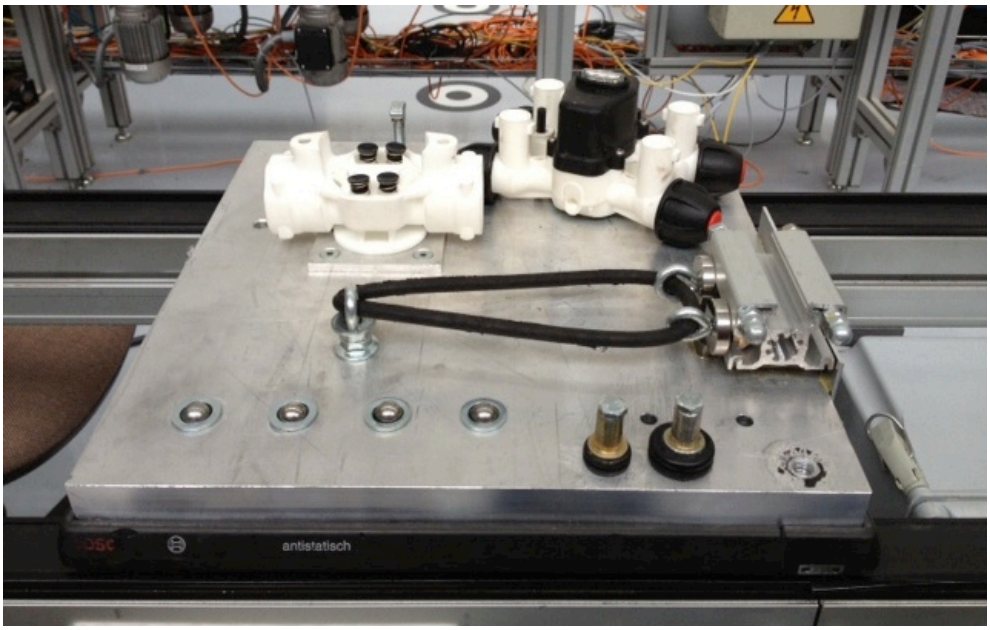


Figure 57: final distribution of the tray

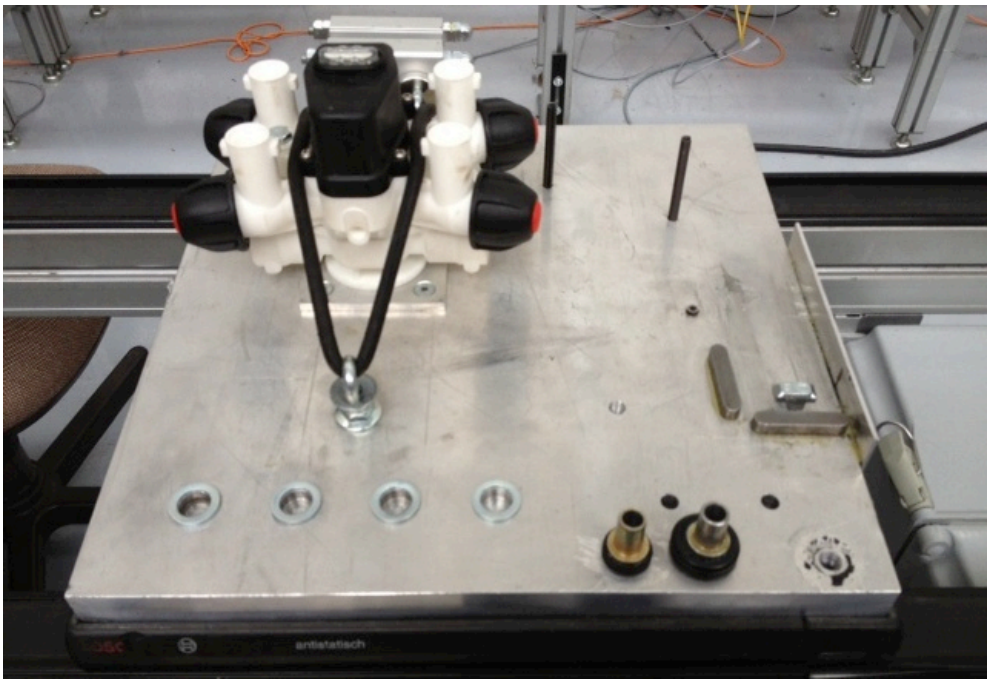


Figure 58: distribution after the assembly operation

5. SOFTWARE FOR THE NEW ASSEMBLY OPERATION

In this chapter 5, everything related to the programming of the PLC and the robot Kawasaki will be explained. Not only does the programming refer to the motion of the robot, but also to the communication procedure between the robot and the PLC as a client-server TCP communication.

This communication procedure between the PLC and each robot will be performed in a way such as a human dialogue but using bytes. In the end, these bytes will mean numbers. This dialogue will control the start of the work process in each work station as well as it will stop each tray that has to be get to one sector in this same sector and if a tray that must not stop in one sector will go through it without stopping there. Furthermore, these bytes will carry information about the number of steps of the process, the last step performed, the previous/next sector, etc. In the end, only one string variable composed by four bytes is sent in each direction. This has to be a string variable because of the only possibility of the Kawasaki of sending and receiving data in ASCII code. The dialogue will be performed with these four bytes sent from the PLC to the robot and another four sent from the robot to the PLC. The use of each byte is going to be explained next:

Bytes sent from the PLC to the ROBOT:

1. The first byte is used for the “handshake”.
2. The second shows the destination sector.
3. The third informs the robot about the first step that has to be performed.
4. The fourth informs the total number of steps of the process.

Bytes sent from the ROBOT to the PLC:

1. The first byte is used for the “handshake”.
2. The second informs the next sector where the tray should be redirected in order to continue with the work.
3. The next step to be performed (it will be 6 if the process is complete).
4. The fourth informs the total number of steps.

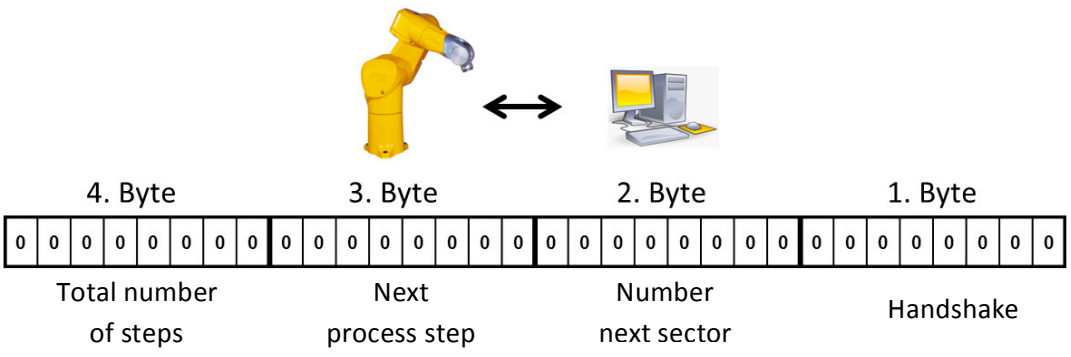


Figure 59: communication PLC-robot

The last three bytes from each side are understandable but the handshake process must be explained more in detail so that the whole process might be understood. Using the first byte from/to each side will perform this handshake. As it was said before, the handshake itself is performed by one dialogue between the PLC and the robot but using numbers. These numbers will vary depending on the sector. Each robot in the laboratory has an own sector number. In this way, these numbers will be 10, 11, 12 and 13 for sector number 1; 70, 71, 72 and 73 for sector number 7 and so on.

The Kawasaki robot is located in sector number 2 so this dialogue will be done with 20, 21, 22 and 23. Each of these numbers has a specific purpose and will communicate a specific message from the robot to the PLC and vice versa.

Before continue explaining, it is important to know how the PLC knows that the tray has to stop in one work station or another. This process is shown in figure 60:

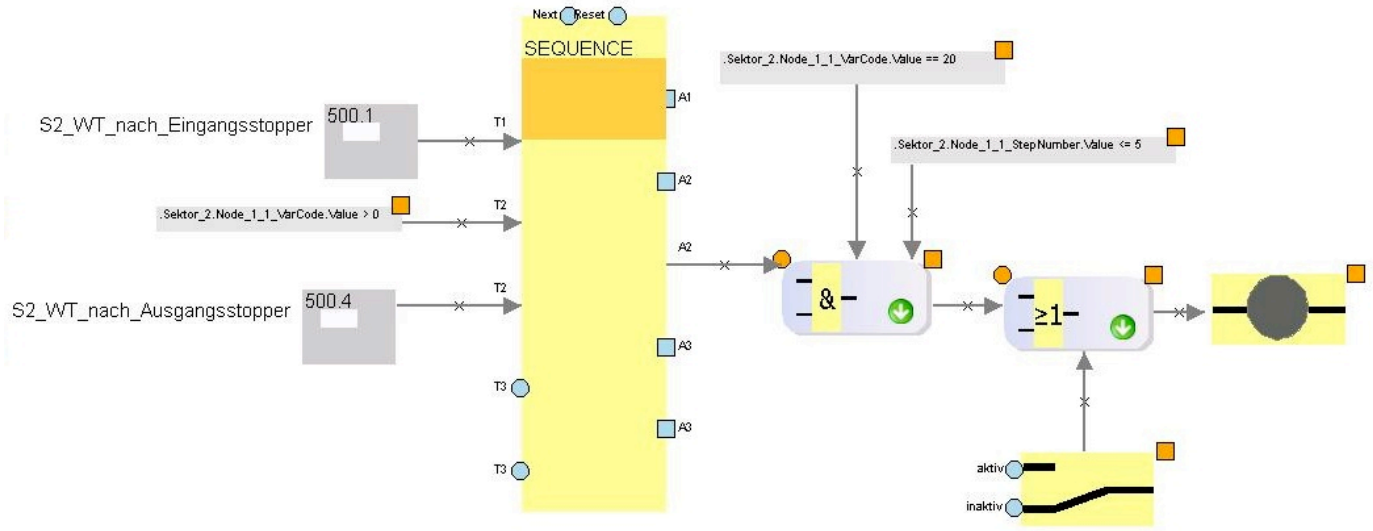


Figure 60: stop process

After the tray has gone through the sensor after the entrance, the next step is to read data. If the data is read, the value of the function of T2 will be greater than 1 and the output A2 will turn ON. The next step is the AND function. A2 is already ON and the other two functions are:

- ".Sektor_2.Node_1_1_VarCode.Value == 20": The value 20 is written in the memory of the tray in order to go to sector 2.
- ".Sektor_2.Node_1_1StepNumber.Value <= 5":The first step to perform is less than 5 (6 meant that the process had been completed).

If these two conditions are met, then the last switch will turn ON turning green and the work station will be able to go up. Once the work station has risen, the PLC will start the handshake with the robot sending the number 20. When the robot reads this 20, it will know that it can start working and will send back 21 so that the PLC knows that the robot has read

his message and it cans delete the 20. When the PLC will send 0, then the robot will delete the 21.

When the robot would finish its work or an error occurred and the work cannot be completed, the robot will send 22 to the PLC so that the PLC can send a message to the work station and it will go down. Once the PLC has read the 22 and the station is down, the PLC will send back 23 in order to let the robot know that it has read its message and the robot can delete the 22. When the PLC reads the 0 again, it will delete the 23. So the possibilities for the PLC are to send 0 (zero, when nothing occurs), 20 (to say “hello, you can start working”) and 23 (to say “I know you have finished or an error has occurred, the tray will leave the work station”) and, on the other hand, the robot can send 0 (zero, if nothing occurs), 21 (to say “Hello, I know I can start working”) and 22 (to say “I have finished my work or an error has occurred”).

The inclusion of the number 23 was done in this master’s thesis. Previously, the communication was performed only with 20, 21 and 22. With only these three messages, the robot could not know whether the PLC had received the message that it had finished its work nor when it could delete this message. Thus, it was thought that a new byte should be included in order to complete the communication and not give the possibility of an error.

The byte for the handshake will be automatically written from the PLC as it will be shown later and the byte from the robot to the PLC will also automatically written inside the motion program of the robot.

This communication might be easier to understand with the next diagram in figure 61. It shows the communication between the PLC and the robot using this language of 0, 20, 21, 22 and 23.

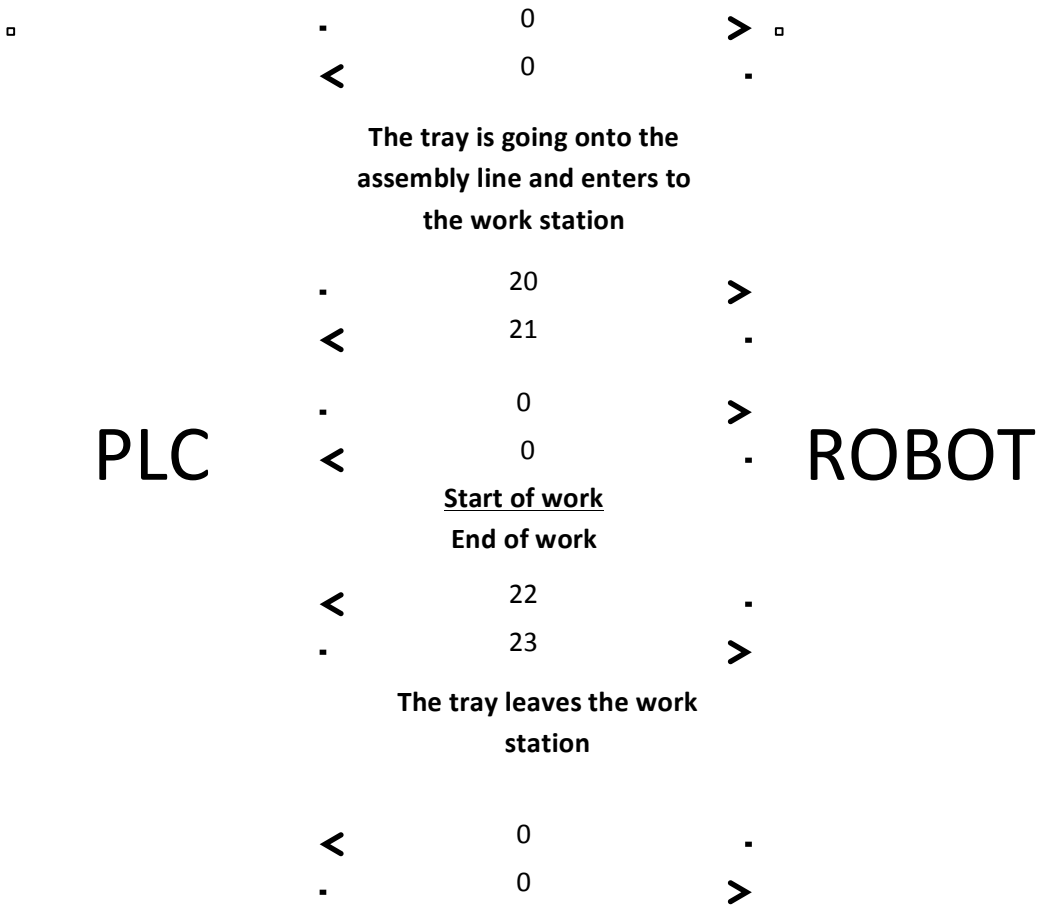


Figure 61: diagram of communication PLC-robot

5.1. PROGRAMMING OF THE KAWASAKI

5.1.1. PROGRAMMING OF THE KAWASAKI AS THE SERVER SIDE

The purpose of this PC-program for the Kawasaki is to establish the connection between the PLC and the robot, to open the socket, to show the status of this socket and to send and receive the string variable flowing in each direction. This variable will be separated into 4 new ones inside this program in order to handle these more easily throughout both this pc program and the motion program.

The code of the Kawasaki for the communication is divided into subprograms or subroutines to make it easier to understand. This subprograms include a own program for starting to wait for an incoming connection from the PLC, after it there is another one to accept this incoming connection, another two for receiving and sending the information sent from the PLC and the information sent from the robot to the PLC and the last two programs for closing the connection and to end waiting for an incoming connection from the PLC or client side.

The best way to explain the whole code is to read through it at the same time that it is explained. At the beginning, the main program will be listed and after it there will be the rest of subprograms or subroutines in order of appearance. The complete code will be at the end of the chapter. Now only the most important and essential parts of it will be explained.

All the commands or instructions such as IFPLABEL, IFPWPRINT… will be explained afterwards when the new interface panel will be explained because these are used to make changes in the interface panel display. The main program is named ***kwmhs.server***. Furthermore, it will be copied with the name AUTOSTART.PC so that it will start automatically when the robot's controller is turned on when its switch "AUTOSTART.PC" is turned on.

.PROGRAM kwmhs.server()

```
mhs_port = 9002
ipplc[0] = 192
ipplc[1] = 168
ipplc[2] = 0
ipplc[3] = 1
```

Information required for the connection with the PLC (port of connection and IP address of the PLC).

```
CALL plc_listen
```

-> Subprogram to start waiting for the incoming connection. The robot starts waiting for the incoming request of connection

```
IF kwretli<0 THEN
  GOTO exit100
END
```

If there is a problem with the subprogram plc_listen, it will guide the main program to close the connection.

```
accept:
  CALL plc_accept
```

Subprogram to accept the connection. The robot will accept the connection required from the PLC and will open a socket.

```
  IF kwaccept<0 THEN
    GOTO exit100
  END
```

If there is a problem with the subprogram plc_accept, it will direct the main program to close the connection.

```
  mhs_repeatloop = TRUE
  WHILE mhs_repeatloop DO
    kwretrec = 0
```

This loop will receive and send the string variable until the connection is finished.

```
  CALL plc_receive ->
```

Subprogram to receive the string variable from the PLC.

```
  CASE kwretrec OF
     VALUE 0:
      mhs_recvcount = mhs_recvcount+1
      $mhs_recvdata = $in_str[0]
```

-> Counter of times data has been received.

```
      IF lengthinstr>10 THEN
        mhs_recvp1 = VAL($MID($mhs_recvdata,1,3))
        mhs_recvp2 = VAL($MID($mhs_recvdata,6,3))
        mhs_recvp3 = VAL($MID($mhs_recvdata,11,3))
        mhs_recvp4 = VAL($MID($mhs_recvdata,16,3))
      END
```

Divide the incoming string variable into 4 different variables.

```
      $out_str1 = $ENCODE(/D,mhs_sendp1)
      $out_str2 = $ENCODE(/D,mhs_sendp2)
      $out_str3 = $ENCODE(/D,mhs_sendp3)
      $out_str4 = $ENCODE(/D,mhs_sendp4)
      $out_strg1 = $out_str1+";"+$out_str2+";
           "+$out_str3+";"+$out_str4
      $send_str[0] = $out_strg1
```

Join the four data variables into only one string variable to be send and it is named

```
      CALL plc_send(kwretsend,$mhs_senddata)
```

-> Subprogram to send the string variable to the PLC.

```
     ANY :
      TYPE "Error during RECEIVE:",kwretrec
      GOTO accept
  END
```

If there is an error in the receiving process, it will have been because of the connection and will redirect the program in order to reaccept this connection.

```
  IF SIG(2101) THEN
      mhs_repeatloop = FALSE
    END
```

Option to exit from the loop from the interface panel.

```
    IF SIG(2104) THEN
      CALL plc_close
      GOTO accept
    END
  END
```

If the button "EXIT" in the interface panel screen has been pushed, it will redirect the program so that the connection will end.

```
exit100:
  CALL plc_close
```

-> Subprogram to close the socket.

```
exit200:
  CALL plc_endlisten
```

-> Subprogram to end the communication.

```
.END (of kwmhs.server)
```

After the main program, the first subprogram that is executed in it is ***plc_listen***. With this subprogram, the robot starts waiting for an incoming connection from the PLC at the programmed port. This port is selected in the main program. In this case it is port 9002.

```
.PROGRAM plc_listen()
  er_count_li = 0
  kwlisten = FALSE
plclisten:
  TCP_LISTEN kwretli,mhsport
  IF kwretli<0 THEN
    kwer_count_li = kwer_count_li+1
    IFPLABEL 15,"LISTEN","Error"
    GOTO plclisten
  ELSE
    IFPLABEL 15,"LISTEN","OK"
  END
.END (of plc_listen)
```

Then, after the PLC tries to connect, the next subprogram executed is ***plc_accept***. It will accept the connection and will open the socket.

```
.PROGRAM plc_accept()#402
  kwer_ac = 0
accept_1:
  IF SIG(2101) THEN
    RETURN
  END
  TCP_ACCEPT kwaccept,mhs_port,toutaccept,ipplc[0]
  IF kwaccept<0 THEN
    kwer_ac = kwer_ac+1
    IFPLABEL 16,"ACCEPT","Error"
    IFPWPRINT 1,1,1,9,10="Trying to connect."
    IFPLABEL 22,,"-"
    GOTO accept_1
  ELSE
    IFPLABEL 16,"ACCEPT","OK"
    IFPWPRINT 1,1,1,9,10="Connected."
    IFPLABEL 22,,"-"
  END
.END (of plc_accept)
```

After the socket has been created, the next step would be to receive and send the information required. For these two steps the subprograms ***plc_receive*** and ***plc_send*** are executed.

```
.PROGRAM plc_receive()
  .numin = 1
  TCP_RECV kwretrec,kwaccept,$in_str[0],.numin,toutrec,mhs_max
  IF kwretrec<0 THEN
    IFPLABEL 1,"Recv Stat","Error"
  ELSE
    IF lengthinstr>0 THEN
      IFPLABEL 1,"Recv Stat","OK"
      mhs_sendcount = mhs_sendcount+1
    END
  END
.END (of plc_receive)

.PROGRAM plc_send(.kwretsend,.$mhs_senddata)
  TCP_SEND kwretsend,kwaccept,$send_str[0],1,toutsend
  IF kwretsend<0 THEN
    IFPLABEL 8,"Send Stat","Error"
  ELSE
    IFPLABEL 8,"Send Stat","OK"
  END
.END (of plc_send)
```

When the connection is wanted to be closed, this program will close the opened socket through where the data is flowing.

```
.PROGRAM plc_close()
  TCP_CLOSE kwretclo,kwaccept
  IF kwretclo<0 THEN
    IFPLABEL 22,,"CLOSE","Error"
    TCP_CLOSE kwret1clo,kwaccept
    IF kwret1clo<0 THEN
    END
  ELSE
    IFPLABEL 22,"CLOSE","OK"
  END
.END (of plc_close)
```

When the socket has been closed, the last step must be to leave the port free. Otherwise, the port will be occupied when the next connection would be performed and an error will appear informing that the port is busy.

```
.PROGRAM plc_endlisten()
  TCP_END_LISTEN kwretendlist,mhs_port
  IF kwretendlist<0 THEN
    IFPLABEL 23,"ENDLISTEN","ERROR"
  ELSE
    IFPLABEL 23,"ENDLISTEN","OK"
  END
.END (of plc_endlisten)
```

After these steps, all the communication is ended and the program could be executed again without any problem. If the main program is stopped or an error occurs and the subprogram ***plc_endlisten*** would not be executed, the pc program ***endlisten*** must be executed manually with the command ***PCEXECUTE endlisten***. It will set free the port 9002. If another port is wanted to set free, the program must be edited.

In summary, the aim of this pc program is to establish the communication between the PLC and the robot and to send and receive the one string variable in each direction. The values of these four bytes of the string variable will not be changed within this program but from the control program using global variables (variables that can be modified from every program).

5.1.2 PROGRAMMING OF THE KAWASAKI FOR THE MOTION

As the main aim of this work is to control the robot from the PLC, this motion program has to start automatically as well. There is a problem: a motion program cannot be executed automatically when the robot is turned on. To solve this problem, there was a need to create a second pc-program from which the motion program will be executed.

This second pc-program will be named AUTOSTART2.PC and its switch will be turned on so that it starts automatically when the robot is turned on. The code of this program is shown next.

```
.PROGRAM autostart2.pc
  IF SWITCH(POWER ) THEN
    GOTO start
  ELSE
    TYPE "PLEASE, TURN POWER ON"
    IFPWPRINT 2,1,1,2,10="Turn POWER ON"
    WAIT SWITCH(POWER )
    IFPWPRINT 2,1,1,9,10=""
  END
start:
  MC ex kw_asop
.END
```

As a motion program is going to be executed, the first requisite is that the motor of the robot must be turned on as otherwise an error will appear. If it is turned off when the program starts, it will show a message in window number 2 in the interface panel screen saying “Turn POWER ON” and the program will wait until it is turned on. Then, the motion program named “**kw_asop**” (assembly operation) will be executed. If a motion program wants to be executed from a pc-program, it must be done by using the command “MC EXECUTE” followed by the name of the program as it is done above.

This program will control the motion of the robot. This will also write the bytes sent to the PLC using the pc-program described above and will perform the handshake. As this assembly operation has five main steps, the motion program will have five different subroutines in order to simplify the program making it easier to understand.

The main program will start initializing some variables to 0 (zero). After it, the loop using WHILE and the variable “**kw_motion”** will be used so that the program is executed time after time. Afterwards, the program will wait in order to receive the first byte containing the message 20 from the PLC. Then it will write 21 in the first byte to send and will wait the 0 from the PLC. After the 0 from the PLC has been read, the robot will send back 0 and the motion program will start.

```
.PROGRAM kw_asop()
  kw_motion = 1
  WHILE kw_motion DO
    mhs_sendp2 = 0
```

```
    mhs_sendp3 = 0
    WAIT mhs_recvp1==20
    mhs_sendp1 = 21
    WAIT mhs_recvp1==0
    WAIT 1
    mhs_sendp1 = 0
```

As there is the possibility of starting in each step independently, there is a CASE __ OF instruction that will start the motion program depending on the third byte coming from the PLC program through the RFID system. The variable **mhs_recvp3,** in which the step to be performed instructed from the PLC is saved, will be copied with the name **kw_firststep.** Depending on this value, the program will go to one of the labels named step1, step2 , … , step 5.

```
    kw_firststep = mhs_recvp3
    CASE kw_firststep OF
     VALUE 1:
      GOTO step1
     VALUE 2:
      GOTO step2
     VALUE 3:
      GOTO step3
     VALUE 4:
      GOTO step4
     VALUE 5:
      GOTO step5
     ANY :
      GOTO exit
    END
```

As there is a specific subroutine for each step, each subroutine will be called from the main program. Its name will be **kw_balls**, **kw_part1**, **kw_elastic**, **kw_screw1**, and **kw_screw2**. Inside each subroutine, there will be a global variable named kw_st1 for the first step and so on. This variable will indicate that the whole step has been performed and consequently that the number of the next step can be sent to the PLC in order to be performed next. When the whole step will be finished, it will take the value 1 and otherwise it will take the value 0. Once the variable has been sent to the variable of each step will be initialized to 0 again.

```
step1:
    mhs_sendp3 = 1
    CALL kw_balls
    WAIT kw_st1==1
    IF kw_st1==1 THEN
      mhs_sendp3 = 2
    END
    kw_st1 = 0
step2:
    mhs_sendp3 = 2
    CALL kw_part1
```

```
    WAIT kw_st2==1
    IF kw_st2==1 THEN
      mhs_sendp3 = 3
    END
    kw_st2 = 0
step3:
    mhs_sendp3 = 3
    CALL kw_elastic
    WAIT kw_st3==1
    IF kw_st3==1 THEN
      mhs_sendp3 = 4
    END
    kw_st3 = 0
step4:
    mhs_sendp3 = 4
    CALL kw_screw1
    WAIT kw_st4==1
    IF kw_st4==1 THEN
      mhs_sendp3 = 5
    END
    kw_st4 = 0
step5:
    mhs_sendp3 = 5
    CALL kw_screw2
    WAIT kw_st5==1
    IF kw_st5==1 THEN
      mhs_sendp3 = 6
    END
    kw_st5 = 0
exit:
    mhs_sendp3 = 6
```

When all the steps have been performed, the number 6 will be sent in order to know that the process has been completed. The next step is to send the tray to the next sector. It is not done automatically so the only idea was to introduce the wanted sector manually form the keyboard, as it is not possible doing it form the touch panel. The next list will be written on the screen and the wanted sector must be inserted and press the key ENTER. The possible destinations are the sector where a work station is located (sector 2, 3, 4, 7, 9 and 11). So if a different number from 20, 30, 40, 50, 70, 90 or 110 is introduced, a message saying “Invalid sector number, please insert it again” will appear. If the number is valid, it will be copied into the second byte and it will be sent to the PLC.

```
    TYPE "SELECT NEXT SECTOR BY TYPING THE NUMBER NEXT TO IT:"
    TYPE "SECTOR 2 (Kawasaki) -> 20"
    TYPE "SECTOR 3 (Adept) -> 30"
    TYPE "SECTOR 4 (Ibm) -> 40"
    TYPE "SECTOR 7 (Staeubli 1) -> 70"
    TYPE "SECTOR 9 (Staeubli 2) -> 90"
```

```
    TYPE "SECTOR 11 (Kuka) -> 110"
    PROMPT "TYPE NUMBER AND PRESS ENTER > ",mhs_nextsector
    CASE mhs_nextsector OF
     VALUE 20,30,40,70,90,110:
      mhs_sendp2 = mhs_nextsector
      kw_destsect = mhs_nextsector/10
      TYPE "DESTINATION SECTOR SELECTED: SECTOR ",kw_destsect
      GOTO next
     ANY :
      GOTO insertsectornum
      TYPE "INVALID SECTOR NUMBER, PLEASE INSERT IT AGAIN"
    END
```

After the work has been finished, the next step is the handshake. The robot will write 22 in the first byte to send to the PLC and will wait to receive the 23 coming from the PLC. Once it reads the 23, it will send back 0 and will wait for the 0 of the PLC.

```
next:
    mhs_sendp1 = 22
    WAIT mhs_recvp1==23
    WAIT 1
    mhs_sendp1 = 0
    WAIT mhs_recvp1==0
  END
.END
```

After the main program has been explained, the subroutines for each step will be explained. It is no necessary to explain line by line as the movements of the robot has been already explained in chapter 4. Each subroutine will use its own position variables. These variables have a common name so that the variables used in step 1 are named kw_01_ii meaning “ii” the order of each point (01 means point 1 and 10 means point 10). The first point of each step will be a joint displacement variable (#kw_01_01, #kw_02_01,…) in order to get not only the position of the tool centre point but also the orientation of all the six joints so that it is always orientated in the same way. The whole code of the motion program including all the subroutines and all the variables can be found at the end of this chapter.

For further information about all the motion subprograms, see annex 1.

5.2. PROGRAMMING OF THE PLC AS THE CLIENT SIDE

5.2.1. MAIN COMMANDS OF OpenMHS

In the next figure 62 some commands and the logic operators are shown. Most of these commands will be used in the program so it is necessary to identify each one before the program is explained.

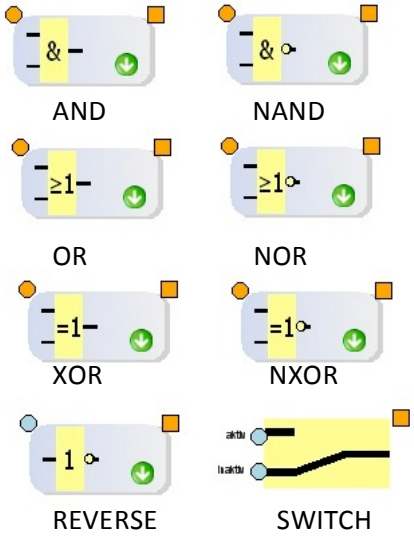


Figure 62: logic operators of OpenMHS

In figure 63, a step sequence can be found. It works as follows: when one input bus is ON, the next output bus turns ON automatically. For example, in figure 63 if the input signal T1 turns ON, signal A2 turns ON. On the other hand, if the next output is not connected to anything, the next input signal must turn ON in order to continue with the sequence.

5.2.2. CONTROLLING THE WORK STATION

Two students from the Hochschule Osnabrück did they "Bachelor Arbeit" developing the communication between the PLC and all the items installed on the assembly line such as sensors, stoppers, robots, etcetera and everything was done but for the Kawasaki station in sector 2.

Their PLC program was taken and used as the base in order to complete the own program for the Kawasaki's work station. The only thing that was done until that point was the set up of all the sensors and one table showing all the internal signal numbers for these sensors (see table 3).

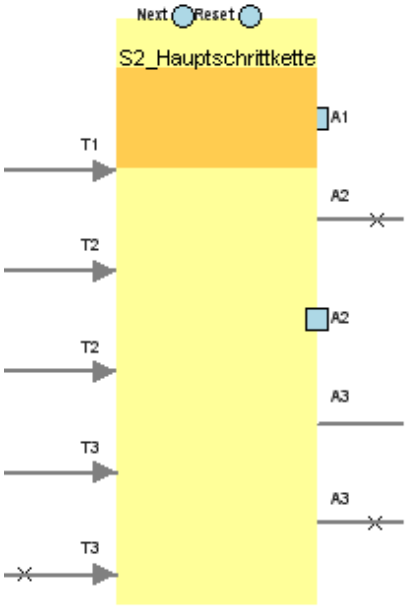


Figure 63: step sequence process

Table 3: input and output signals

1.Work station / Sector 2	MHS	Card Nr.:	Slot:	Testprogramm Nr.:	Terminal:	Labelling:
INPUTS						
S2_Before_Entrance_Stopper	500.0	5	D-IN0	IN1	E0.0	E2.0
S2_After_Entrance_Stopper	500.1	5	D-IN1	IN2	E0.1	E2.1
S2_Station UP	501.6	5	D-IN14	IN15	E1.6	E2.2
S2_Station DOWN	501.5	5	D-IN13	IN14	E1.5	E2.3
S2_Before_Exit_Stopper	500.3	5	D-IN3	IN4	E0.3	E2.4
S2_After_Exit_Stopper	500.4	5	D-IN4	IN5	E0.4	E2.5
OUTPUTS						
S2_Entrance_Stopper_lower	500.0	5	D-OUT0	OUT1	A0.0	A2.0
S2_Station_rise	500.7	5	D-OUT7	OUT8	A0.7	A2.1
S2_Station_lower	500.2	5	D-OUT2	OUT3	A0.2	A2.2
S2_Exit_Stopper_lower	500.1	5	D-OUT1	OUT2	A0.1	A2.3

As it has been said before, each work station has two stoppers: one at the entrance and a second one at the exit. Before and after each stopper a sensor is located so that the PLC can check where the trays are. With this four sensor and another one to check whether the work station is up or down, the PLC can control the whole station.

In figure 64, the whole programming diagram for the control of the work station is shown. As it might be confused, every step will be explained one by one. The main step sequence will be taken as the reference to explain everything so from now on if a T1, T2… or A1, A2… are mentioned, these will do reference to the input and output signals or buses of it as it can bee seen in figure 63.

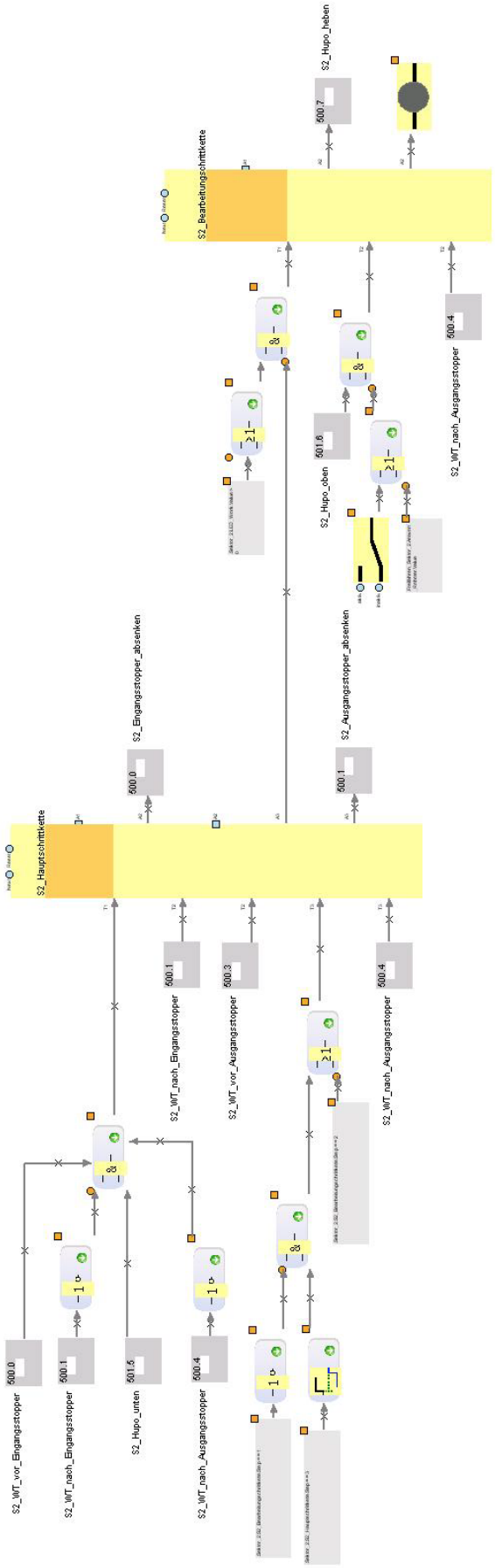


Figure 64: control program

The first signal incoming (T1) will be ON when the next four conditions would be satisfied:

- The sensor before the entrance stopper is ON (input signal 500.0).
- The sensor after the entrance stopper is OFF (input signal 500.1).
- The work station is DOWN (input signal 501.5).
- The sensor after the exit stopper is ON (input signal 500.4).

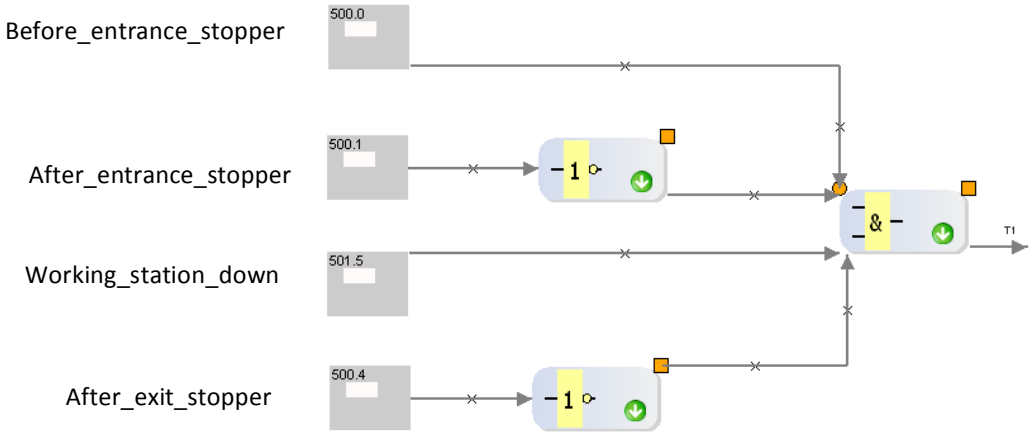


Figure 65: T1

Then, because of the structure of a step sequence, output A2 will be ON turning ON the output signal 500.0. This signal will lower the entrance stopper allowing the tray to go onto the work station.

Figure 66: A2

Before the next output of the main step sequence, two input signals must turn ON one after the other. The first one is input signal 500.1 (the sensor after the entrance will turn ON, T2.1) and the second is input signal 500.3 (the sensor before the exit stopper will turn ON, T2.2). This will mean that the tray is in position and the work station can rise if this try must stay there.

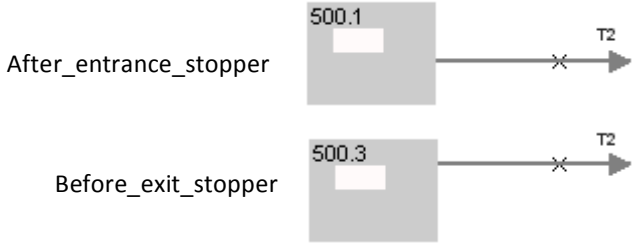


Figure 67: T2.1 and T2.2

After the signal T2.2 is ON, the signal A3.1 will turn ON as well. This A3.1 signal goes to the processing step sequence start. This sequence will be explained afterward, after the main step sequence is totally explained.

The next input is T3.1. As it can be seen, the signal T3.1 will turn ON if one of the following terms is met:

- If both the step 1 of the processing step sequence is OFF and the step 3 is ON (it means when A2 turns OFF and the sensor after the exit is ON).
- If the work station is UP and the robot sends the message 22 that means it has finished.

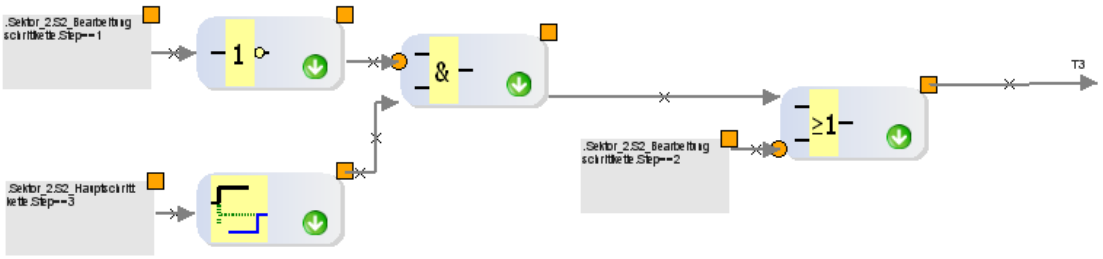


Figure 68: T3

When signal T3.1 is ON, the next output A3.2 would turn ON turning ON output signal 500.1. This means that the work station will lower.

Figure 69: A3.2

To complete the whole sequence, the last input signal 500.4 must turn ON (the sensor after the exit stopper turns ON, T3). This would finish the sequence and it would start again waiting for the first conditions to meet.

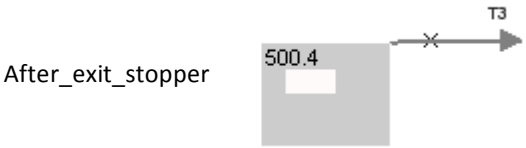


Figure 70: T3.2

The second sequence of the program is the processing step sequence. An image of the whole sequence can be found in figure 71.

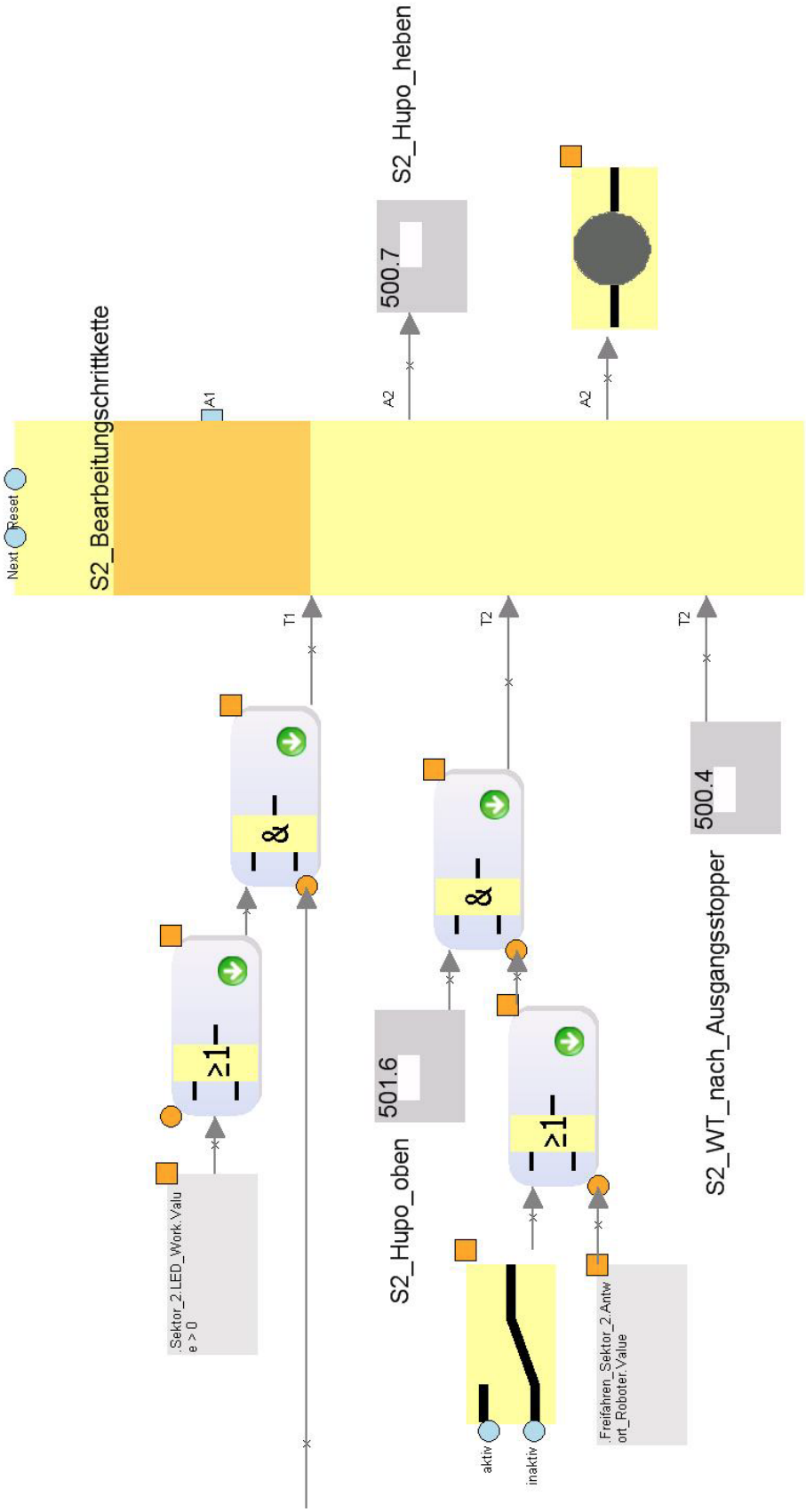


Figure 71: work station up/down control program of PLC

When the signal A3 from the main step sequence is ON and the work process explained at the beginning of this chapter is ON (see figure 60), the signal A2.1 will turn ON turning ON the output A2 and turning ON output signal 500.7 (work station will rise taking the tray up).

When the station is UP, input signal 501.6 should turn ON. Then, there are two possibilities to make the station go down. When the robot will send the number 22 (it has finished its work) or the switch is pressed manually. If one of these two options is met, signal T2.1 will turn on, making A2.2 also ON and lowering the station. The tray will leave the work station and the sensor after the exit stopper (input signal 500.4) will turn on finishing this process.

In summary, and to clarify a bit more the whole process, it will be describe more simply with other words.

When the tray reaches the sensor before the entrance stopper, and if there is not a tray on the work station, the stopper will go down allowing the tray to pass it and then it will rise again when the tray goes on the sensor after the entrance stopper. Once the tray reaches the sensor before the exit stopper there are two options:

- First option: if this tray has to stop in this station, the station will rise. The PLC will send 20 to the robot, the robot will answer 21 and will work on the tray. When it finishes or an error occurs, the robot will send 22 and the PLC will send 23 and it will lower the station.
- Second option: if it does not have to stop in this station the station will not raise.

After one option or the other, the PLC will lower the exit stopper allowing the tray to leave the station. When the tray goes onto the sensor after the exit stopper will rise again.

This is the whole process simplified.

5.2.3. HANDSHAKE AND COMMUNICATION WITH THE FOUR BYTES – PLC

After all aspects about the work station have been explained, the next step is to explain how the PLC sends the four variables automatically to the robot. It must be remembered that four bytes are used and that the first one was for the handshake. A step sequence process has been also used for this process as shown in figure 72.

The process of the handshake has already been explained but as its figure can be confused, each step will be described independently. The names of the inputs are not the name in the image but the step (meaning input 1 the first input).

The whole programming can be seen in next figure 72. As it could be not clear enough, a figure for each step will be shown individually.

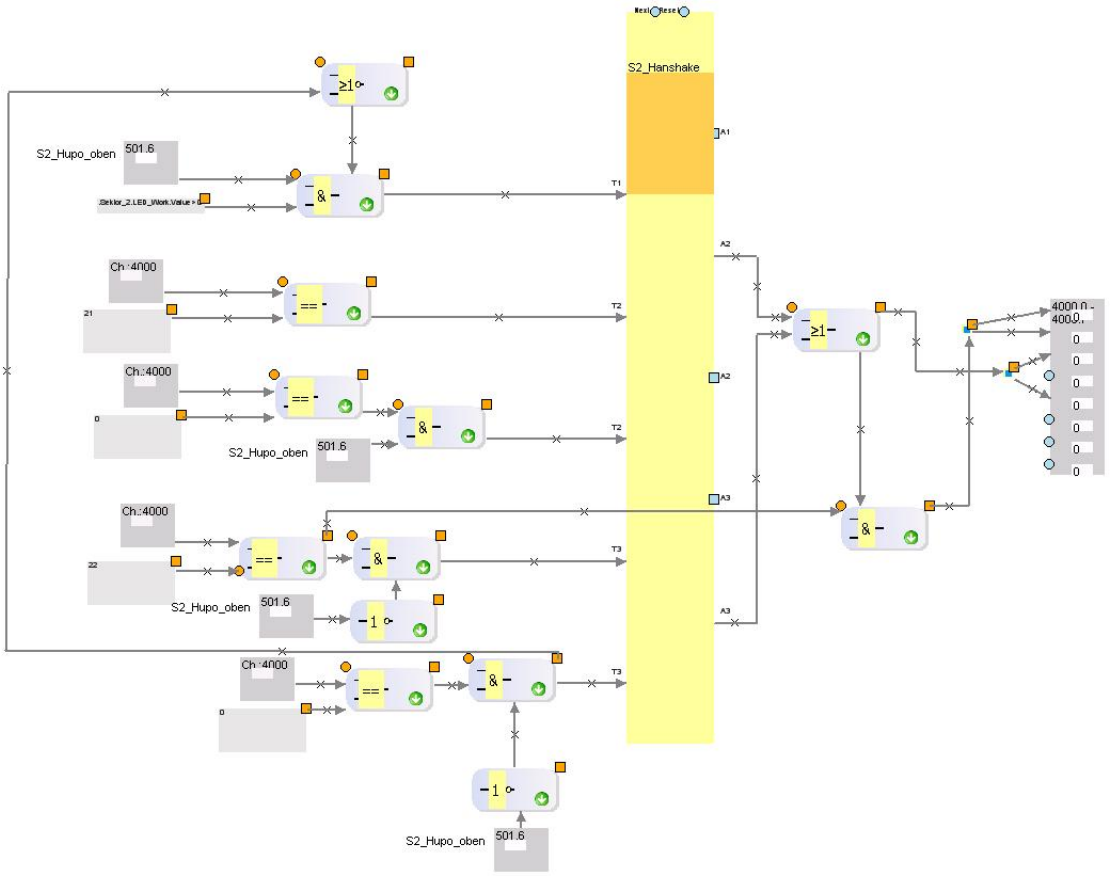


Figure 72: handshake

Input 1 (T1): the following conditions must meet (see figure 73):

- Signal 501.6 in ON (the work station is UP).
- The step T1 from the process step sequence must be ON.
- The step 5 of this process must be ON (to check that the tray is coming and the robot is not working as there are two possibilities of receiving the number 0 from the robot: when there is no tray on the work station and when the robot is working).

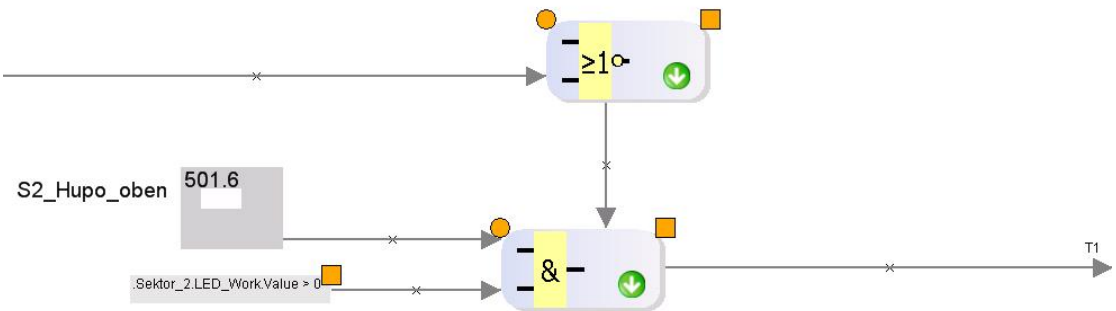


Figure 73: input 1

Output 1 (A2): when input 1 is ON, output 1 is turned also ON. It will mean that the bites 3 and 5 turn ON (00101000 -> 4+16=20) so the PLC will send 20 into the first byte to the robot (see figure 74).

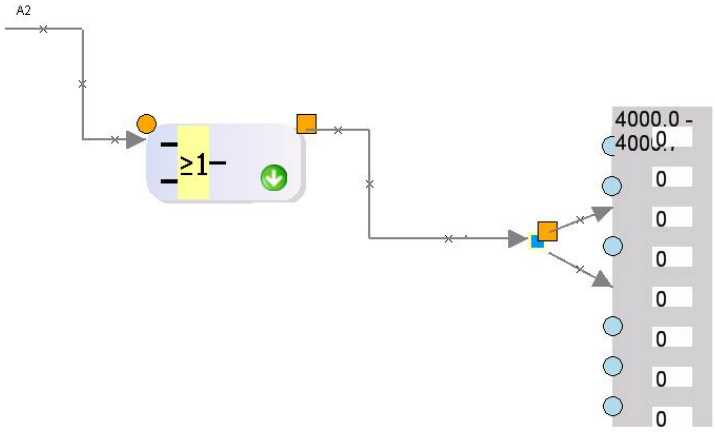


Figure 74: output 1

Input 2 (T2.1): when the robot sends back 21, input 2 turns ON and the 20 sent into the first byte is deleted (see figure 75).

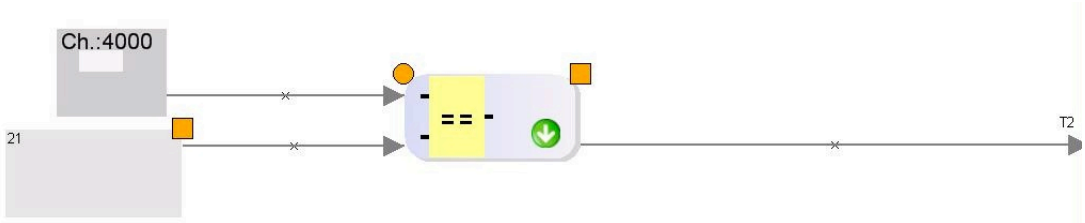


Figure 75: input 2

Input 3 (T2.2): the station is UP and the robot sends 0 again in the handshake byte (see figure 76).

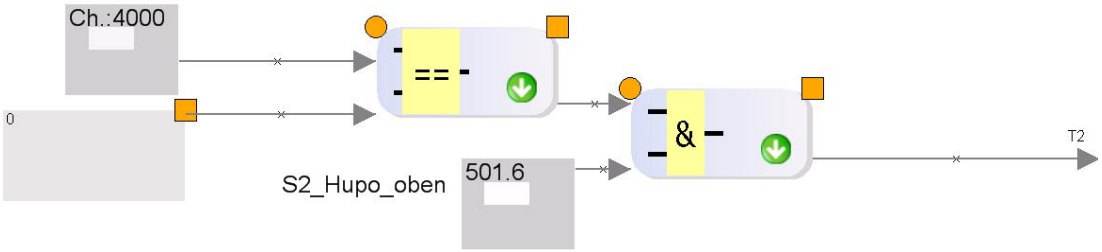


Figure 76: input 3

Input 4 (T3.1): the following conditions must meet (see figure 77).

- The robot sends 22 (means that it has finished its work).
- The signal 501.6 is OFF (means that the station has been taken down)

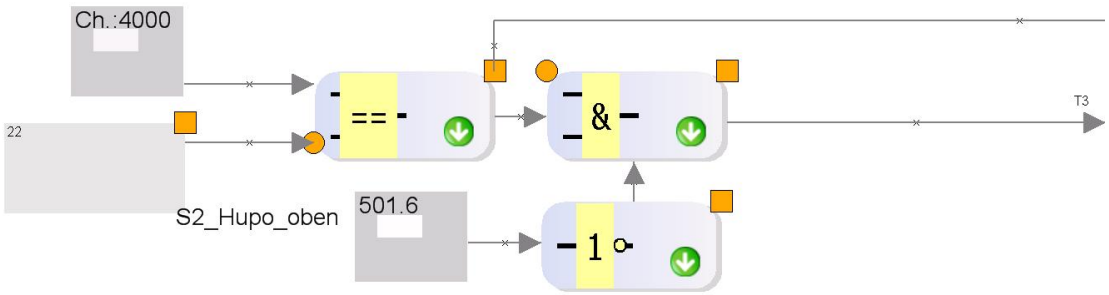


Figure 77: input 4

Output 4 (A3): it happens the same than output 1 but also bites 1 and 2 turn ON (11101000 -> 1+2+4+16=23) so 23 is sent in the first byte to the robot (see figure 78).

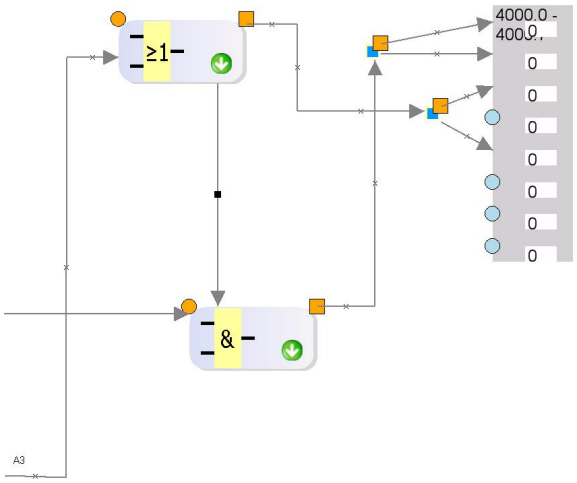


Figure 78: output 4

Input 5 (T3.2): the robot has read the 23 and deletes the 22 sending 0. When it happens, the PLC deletes also the 23 sending 0 (see figure 79).

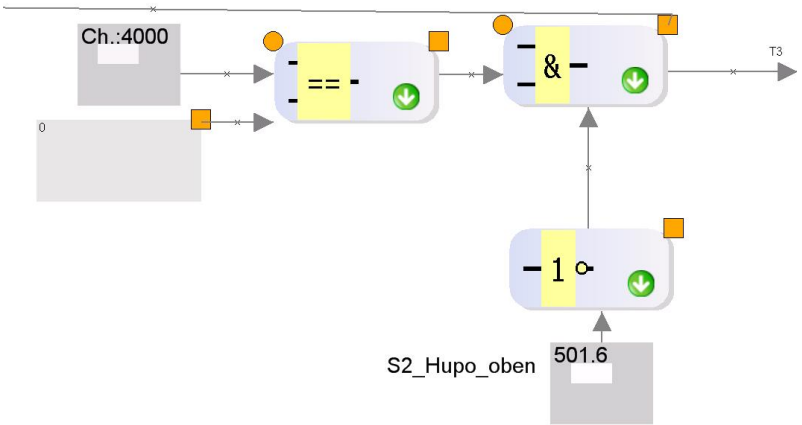


Figure 79: input 5

5.3. NEW INTERFACE PANEL

As it has been said in chapter 2, an interface panel is normally required to operate the robot and peripheral equipment together through a variety of hard switches and lamps. This interface will show all the required information (the information send to the PLC and received from the PLC), the status of the connection with the PLC, etcetera. Each of the switches, lamps, and windows will be explained one by one on the next paragraphs.

First of all, an image of this new interface panel screen will be shown in figure 80 so that it will be easier to follow the explanation.

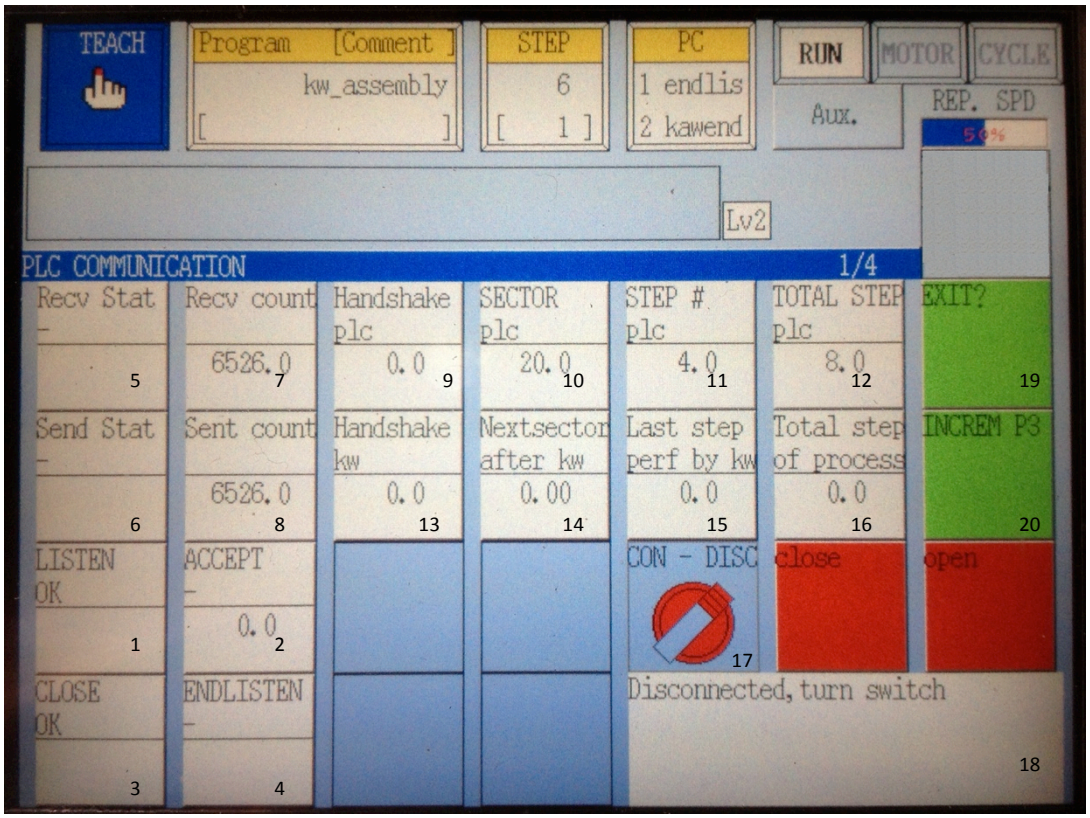


Figure 80: interface panel screen

As it can be seen, there are more or less twenty different windows. Each of these windows will be explained next.

The numbered displays in figure 80 from 1 to 4 are related with the communication status. These will show the status of this communication throughout all the four steps regarding the robot: listen, accept, close and end listen. These four are string data displays.

1. It is named “LISTEN” and will show the status of the listening process of the robot. If the listening subprogram “plc_listen” has been run, an “Ok” label will be shown there. Otherwise, if there has been a problem or it has not been run correctly, an “Error” label will appear.

2. It is named “ACCEPT” and will show the status of the accepting process with the subprogram “plc_accept”. The procedure is the same as in the “LISTEN” display. This window will also display, in case of error, the number of times that the Kawasaki has tried to accept the connection (0 in this case).

3. It is named “CLOSE” and will show the status of the closing process through the “plc_close” subprogram. The procedure is the same as in the “LISTEN” display.

4. It is named “ENDLISTEN” and will show the status of the end-listen process with the subprogram “plc_endlisten”. The procedure is the same as in the “LISTEN” display.

Once that the communication has been displayed the next step is to show the status of the data flow between the PLC and the robot. Displays from 5 to 8 show this status. Displays 5 and 6 are string data displays and 7 and 8 are variable data displays.

5. It is named “Recv Stat” and it will show the status of the receiving data process. If the robot is receiving data “Ok” will appear and if there is an error “Error” will appear. When there is not connection only a label with “-“ is found.

6. It is named “Send Stat” and, in this case, it will show the status of the sending data process. The procedure is the same as in the “Recv Stat” display.

7. It is named “Recv count”. It shows the variable “mhs_recvcount” which counts the number of times that data has been received from the PLC. This variable is controlled as well from the main program.

8. It is named “Send count”. It shows the variable “mhs_sendcount” which counts the number of times that data has been sent to the PLC. This variable is also controlled from the main program.

At this point, it is always right to show the information received and send to check if it is working correctly. As it has been explained before, the communication between the PLC and the robot is based on four bytes or variables. These four variables were related with the handshake, the number of next sector, the last step performed and the total of steps. One variable data display will be used to show each of these variables. These are numbers from 9 to 16 in figure 80. From 9 to 12 are data received from the PLC, and from 13 to 16 are data sent from to the PLC.

9. It will show the “handshake” variable from the PLC. As the Kawasaki works in sector 2, this variable can be 0 or 20 or 23 as it has been explained before.

10. It will show the sector number from where the assembly tray would come from.

11. It will show the first step that has to be performed.

12. It will show the total number of steps of the whole assembly operation.

13. It will show the "handshake" variable from the robot. As it works in sector 2, this variable can be 0 or 21 or 22 as it has been explained before.

14. It will show the number of the sector where the assembly tray should go after this sector. So if after the assembly at the Kawasaki, it should go to sector 7, a 70 will be saved into this variable.

15. It will show the last complete step performed by the Kawasaki.

16. It will show the total number of steps the assembly operation has. If the assembly operation has been performed completely, the variable of the last step completed and this variable should be the same.

The other displays are not showing the status of anything nor information received or sent. These are going to be explained one by one next by its numbers.

17. This is a 2-notch selector switch. If this is turned left signal 2103 is turned on and the connection between the PLC and the robot is allowed whereas if it is turned right, signal 2104 is activated and the connection is closed using the "plc_close" subprogram and the server side stays waiting for signal 2103 again.

18. This is a window display. It shows the status of the connection and the status of the switch shown in point 17.

19. This is a push button. It is named "EXIT?" and if it is pressed, signal 2101 is activated the communication between the PLC and the robot is closed and the server side program is ended.

20. This is another push button. Its name is "INCREM P3" and when it is pressed, signal 2102 is activated and 1 is added to the variable, which shows the last step performed by the robot. This can be useful when an error occurs and the last step cannot be notified from the motion program to the PLC.

21. This window display will show a message saying "TURN POWER ON" when the programs are automatically started and the power is off as it has been explained in subchaprter 5.2.

6. CONCLUSION AND OUTLOOK

The integration of one robot and the assembly of one product allow learning everything about the functionality of a robot. Not only the different possibilities of motion but also the communication between the robot and its surroundings. It has been a complete master's thesis as it included electronics, mechanics and design via CAD.

Before this work, the robot work cell was not correctly designed as when it worked at maximum speed, the whole cell moved, moving the robot and making it impossible to get a proper final accuracy. It did not make sense to have such a robot not being able to work with small cycle times. This problem has been solved and the robot is able to work as fast as the process would allow it.

Furthermore, with the new design of the platform, the whole work cell and the robot could be easily relocated somewhere else in the lab, or even outside of it, in order to integrate a new robot into the assembly line and the PLC.

The inclusion of this robot in the assembly line is the conclusion, for the time being, of the construction of the assembly line started some semesters ago. With this robot, there are already six robots working directly, or with this possibility, on the assembly line. These are two identical Stäubli robots, one Kuka robot, one IBM, one Adept and this Kawasaki robot. Furthermore, all the work stations are occupied but only the one for the Kuka robot is not completely installed.

After this master's thesis, the Kawasaki robot can be used as a server side. It means that the robot can be totally controlled from the PLC. The PLC can instruct the robot what to do. Not only could the robot assemble the mentioned product but also it would only be needed to reprogram the motion program in order to assemble another product. Reprogramming the motion program means to write a new program teaching the new motion points and the new paths to be followed while taking into consideration the control of the global variables used for the communication between the robot and the PLC.

On the other hand, the communication program does not have to be reprogrammed as it only creates the communication and sends and receives one string variable in each direction. The only required aspect is the use of the same global variables (variables with the same name that can be modified from every program) used in the motion program.

In these days, only one motion program has been programmed, as there was only one product to be assembled. In the future, more possibilities of assembly operations might be taken into consideration so that the robot could assembly different products at the same time. The PLC might send a fifth byte in order to let the robot know which assembly operation has to be executed depending on the product that might be located on top of the tray that has reached the work station.

The possibility of working on both the working station and the table, would allow assembling a more complex product, as all the needed parts might be located on the table and not on a

small tray. This would allow having a product with more parts to be assembled. The other possibility would be to work directly on the table and, when the product might be totally assembled, it could be picked up and placed on top of a tray on the working station. From here, the tray could be redirected to another work station another robot could perform more steps and finish the whole work.

For example, the first steps could be realized by the Kawasaki and then the tray could be redirected to the work station of the Kuka. This will allow assembling a really complex product due to the different possibilities that the two robots would have working together.

If the robot might be relocated away from the assembly line, some other ideas could be the assemble of one product by using only the Kawasaki as an isolated robot. It would have to be reprogrammed in order to work without being a server side. Nevertheless, the possibility of moving the whole cell of the Kawasaki away from the assembly line, could allow another robot taking its place and the possibility of being integrated in the assembly line. This would be easily performed and the only step to perform would be to create an appropiate server side program for the PLC as all the electronics and mechanics of the work station are already installed and working properly. While another robot is integrated, the Kawasaki robot can work independently located in another place in the laboratory.

Besides, some improvements could be performed in order to improve the assembly operation. As the product was only one prototype, the assembly process did not have to be a 100% perfect process. The main problems of this process were in the screwing process. As it is almost impossible to manufacture the tool adapter and the hollow cylinder perfectly, the tolerances make it impossible that the joint of this cylinder and the last joint of the robot were coaxial. This make it impossible to performe a perfect screwing. A solution for this problem, in the case that the assembly process would be taken to the real process, would be to manufacture a new tool. This tool might have its own small motor. This motor would rotate the screwing tool solving this problem at once. Furthermore, there were also some problems when this same tool tried to pick up the bolt from its position. The solution for this problem might be a supplier of bolts that would locate the bolts already inside this new tool with its own motor. This would eliminate the problem.

Finally, this master thesis has been the perfect addition to the degree courses of Mechanical Engineering. In the last ten semesters, this is what the degree used to last in Spain, not many aspects about electricity and electronics were studied but basic aspects of these. Everything was new and everything had to be learnt form the very beginning. Even so, with enthusiasm and effort everyone can learn what is needed to achieve the required objectives.

7. REFERENCES

1. Bolton, W: Programmable Logic Controller. 5th edition. Oxford: Elsevier, 2009, page 3.

2. Soft PLC corporation. www.softplc.com. Texas, USA. 01.04.2013.

3. Kawasaki robotics, INC. (Web). www.kawasakirobotics.com. USA. 10.04.2013.

4. Oracle (Web). www.oracle.com/java/socket-140484.html. United Kingdom. 25.04.2013.

5. Kawasaki Heavy Industries, Ltd., Kawasaki Robot Controller E-Series: TCP/IP communication manual. 1st edition. 2010.

6. Kawasaki Heavy Industries, Ltd., Kawasaki Robot Controller E-Series: AS Language Reference Manual. 1st edition. 2010.

8. Stevens, R.W.; Fall, K.R.: TCP/IP Illustrated volume 1: The Protocols. USA: Addison-Wesley, 1993.

7. Kawasaki Heavy Industries, Ltd. Kawasaki Robot Controller E-Series: Operation Manual. 1st edition. 2010.

9. Finkenzeller, Klaus: RFID Handbook, fundamentals and applycations in contactless smart cards and identification. 2nd edition. United Kingdom: Wiley, 2003.

10. Simon, A.X.; Markus, D.: Warenträgersteuerung mittels RFID-Technik. Osnabrück, Germany: Hochschule Osnabrück, 2013.

11. PEPPERL+FUCHS GmbH (Web). http://www.pepperl-fuchs.co.uk/great_britain/en/index.htm. 01.05.2013.

12. Amazone (Web). www.amazone.de. 07.05.2013.

13. Porosky, Peter: Measuring Water. USA: University Press of America, 2007. PublishAmerica.

14. Laugthon, M; Warne, D.: Electrical Engineers's Reference Book. 16th edition. United Kingdom: Newnes, 2003.

8. ANNEX 1: PROGRAMMING CODE

8.1. PROGRAMMING CODE OF THE PC-PROGRAM

```
.PROGRAM kwmhs.server()
  IFPLABEL 22,,"-"
  IFPLABEL 23,,"-"
  mhs_port = 9002
  ipplc[0] = 192
  ipplc[1] = 168
  ipplc[2] = 0
  ipplc[3] = 1
  kwretli = 0
  kwaccept = 0
  toutaccept = 2
  toutrec = 60
  mhs_max = 255
  toutsend = 60
  kwretclo = 0
  kwret1clo = 0
  kwretendlist = 0
  mhs_recvcount = 0
  $mhs_recvdata = "empty"
  mhs_sendcount = 0
  $mhs_senddata = "empty"
  mhs_sendp1 = 0
  mhs_sendp2 = 0
  mhs_sendp3 = 0
  mhs_sendp4 = 0
  mhs_recvp1 = 0
  mhs_recvp2 = 0
  mhs_recvp3 = 0
  mhs_recvp4 = 0
  $out_str1 = ""
  $out_str2 = ""
  $out_str3 = ""
  $out_str4 = " "
  CALL plc_listen
  IF kwretli<0 THEN
    GOTO exit100
  END
accept:
  accept_loop = TRUE
  IF SIG(2104) THEN
    IFPWPRINT 1,1,1,9,10="Disconnected,turn switch"
    IFPLABEL 16,,"-"
    IFPLABEL 1,,"-"
    IFPLABEL 8,,"-"
  END
  WHILE accept_loop DO
    IF SIG(2101) THEN
      GOTO exit200
    END
```

```
    IF SIG(2103) THEN
      accept_loop = FALSE
    END

  END
  CALL plc_accept
  IF kwaccept<0 THEN
    GOTO exit100
  END
  mhs_repeatloop = TRUE
  WHILE mhs_repeatloop DO
    kwretrec = 0
    CALL plc_receive
    CASE kwretrec OF
     VALUE 0:
      mhs_recvcount = mhs_recvcount+1
      $mhs_recvdata = $in_str[0]
      IF lengthinstr>10 THEN
        mhs_recvp1 = VAL($MID($mhs_recvdata,1,3))
        mhs_recvp2 = VAL($MID($mhs_recvdata,6,3))
        mhs_recvp3 = VAL($MID($mhs_recvdata,11,3))
        mhs_recvp4 = VAL($MID($mhs_recvdata,16,3))
      END
      $out_str1 = $ENCODE(/D,mhs_sendp1)
      $out_str2 = $ENCODE(/D,mhs_sendp2)
      $out_str3 = $ENCODE(/D,mhs_sendp3)
      $out_str4 = $ENCODE(/D,mhs_sendp4)
      $out_strg1 =
$out_str1+";"+$out_str2+";"+$out_str3+";"+$out_str4
      $send_str[0] = $out_strg1
      kwretsend = 0
      CALL plc_send(kwretsend,$mhs_senddata)
     ANY :
      TYPE "Error during RECEIVE:",kwretrec
      GOTO accept
    END
    IF SIG(2101) THEN
      mhs_repeatloop = FALSE
    END
    IF SIG(2102) THEN
      mhs_sendp3 = mhs_sendp3+1
    END
    IF SIG(2104) THEN
      CALL plc_close
      GOTO accept
    END
  END
exit100:
  CALL plc_close
exit200:
  CALL plc_endlisten
  mhs_sendcount = 0
  mhs_recvcount = 0
  $mhs_recvdata = "-"
```

```
  $send_str[0] = "-"
  IFPLABEL 1,,"-"
  IFPLABEL 8,,"-"

  IFPLABEL 15,,"-"
  IFPLABEL 16,,"-"
  $out_str1 = "-"
  $out_str2 = "-"
  $out_str3 = "-"
.END



.PROGRAM plc_accept()
  kwer_ac = 0
accept_1:
  IF SIG(2101) THEN
    RETURN
  END
  TCP_ACCEPT kwaccept,mhs_port,toutaccept,ipplc[0]
  IF kwaccept<0 THEN
    kwer_ac = kwer_ac+1
    IFPLABEL 16,"ACCEPT","Error"
    IFPWPRINT 1,1,1,9,10="Trying to connect."
    IFPLABEL 22,,"-"
    GOTO accept_1
  ELSE
    IFPLABEL 16,"ACCEPT","OK"
    IFPWPRINT 1,1,1,9,10="Connected."
    IFPLABEL 22,,"-"
  END
.END



.PROGRAM plc_close()
  TCP_CLOSE kwretclo,kwaccept
  IF kwretclo<0 THEN
    IFPLABEL 22,,"CLOSE","Error"
    TCP_CLOSE kwret1clo,kwaccept
    IF kwret1clo<0 THEN
    END
  ELSE
    IFPLABEL 22,"CLOSE","OK"
  END
.END


.PROGRAM plc_endlisten()
  TCP_END_LISTEN kwretendlist,mhs_port
  IF kwretendlist<0 THEN
    IFPLABEL 23,"ENDLISTEN","ERROR"
  ELSE
    IFPLABEL 23,"ENDLISTEN","OK"
```

```
  END
.END


.PROGRAM plc_listen()
  er_count_li = 0
  kwlisten = FALSE
plclisten:
  TCP_LISTEN kwretli,mhsport
  IF kwretli<0 THEN
    kwer_count_li = kwer_count_li+1
    IFPLABEL 15,"LISTEN","Error"
    GOTO plclisten
  ELSE
    IFPLABEL 15,"LISTEN","OK"
  END
.END


.PROGRAM plc_receive()#416969
  .numin = 1
  TCP_RECV kwretrec,kwaccept,$in_str[0],.numin,toutrec,mhs_max
  IF kwretrec<0 THEN
    IFPLABEL 1,"Recv Stat","Error"
  ELSE
    IF lengthinstr>0 THEN
      IFPLABEL 1,"Recv Stat","OK"
      mhs_sendcount = mhs_sendcount+1
    END
  END
.END


.PROGRAM plc_send(.kwretsend,.$mhs_senddata)#416742
  TCP_SEND kwretsend,kwaccept,$send_str[0],1,toutsend
  IF kwretsend<0 THEN
    IFPLABEL 8,"Send Stat","Error"
  ELSE
    IFPLABEL 8,"Send Stat","OK"
  END
.END
```

8.2. PROGRAMMING CODE OF THE MOTION PROGRAM

```
.PROGRAM autostart2.pc()
  IF SWITCH(POWER ) THEN
    GOTO start
  ELSE
    TYPE "PLEASE, TURN POWER ON"
    IFPWPRINT 2,1,1,2,10="Turn POWER ON"
    WAIT SWITCH(POWER )
    IFPWPRINT 2,1,1,9,10=""
  END
start:
  MC ex kw_asop
.END



.PROGRAM kw_asop()
  kw_motion = 1
  WHILE kw_motion DO
    mhs_sendp2 = 0
    mhs_sendp3 = 0
    WAIT mhs_recvp1==20
    mhs_sendp1 = 21
    WAIT mhs_recvp1==0
    WAIT 1
    mhs_sendp1 = 0
    kw_firststep = mhs_recvp3
    mhs_sendp4 = 6
    CASE kw_firststep OF
     VALUE 1:
      GOTO step1
     VALUE 2:
      GOTO step2
     VALUE 3:
      GOTO step3
     VALUE 4:
      GOTO step4
     VALUE 5:
      GOTO step5
     ANY :
      GOTO exit
    END
step1:
    mhs_sendp3 = 1
    CALL kw_balls
    WAIT kw_st1==1
    IF kw_st1==1 THEN
      mhs_sendp3 = 2
    END
    kw_st1 = 0
step2:
    mhs_sendp3 = 2
    CALL kw_part1
```

```
    WAIT kw_st2==1
    IF kw_st2==1 THEN
      mhs_sendp3 = 3
    END
    kw_st2 = 0
step3:
    mhs_sendp3 = 3
    CALL kw_elastic
    WAIT kw_st3==1
    IF kw_st3==1 THEN
      mhs_sendp3 = 4
    END
    kw_st3 = 0
step4:
    mhs_sendp3 = 4
    CALL kw_screw1
    WAIT kw_st4==1
    IF kw_st4==1 THEN
      mhs_sendp3 = 5
    END
    kw_st4 = 0
step5:
    mhs_sendp3 = 5
    CALL kw_screw2
    WAIT kw_st5==1
    IF kw_st5==1 THEN
      mhs_sendp3 = 6
    END
    kw_st5 = 0
exit:
insertsectornum:
    TYPE "SELECT NEXT SECTOR BY TYPING THE NUMBER NEXT TO IT:"
    TYPE "SECTOR 2 (Kawasaki) -> 20"
    TYPE "SECTOR 3 (Adept) -> 30"
    TYPE "SECTOR 4 (Ibm) -> 40"
    TYPE "SECTOR 7 (Staeubli 1) -> 70"
    TYPE "SECTOR 9 (Staeubli 2) -> 90"
    TYPE "SECTOR 11 (Kuka) -> 110"
    PROMPT "TYPE NUMBER AND PRESS ENTER > ",mhs_nextsector
    CASE mhs_nextsector OF
     VALUE 20,30,40,70,90,110:
      mhs_sendp2 = mhs_nextsector
      kw_destsect = mhs_nextsector/10
      TYPE "DESTINATION SECTOR SELECTED: SECTOR ",kw_destsect
      GOTO next
     ANY :
      GOTO insertsectornum
      TYPE "INVALID SECTOR NUMBER, PLEASE INSERT IT AGAIN"
    END
next:
    mhs_sendp1 = 22
    WAIT mhs_recvp1==23
    WAIT 1
    mhs_sendp1 = 0
```

```
    WAIT mhs_recvp1==0
  END
.END


.PROGRAM kw_balls()
  SPEED 100 ALWAYS
  JMOVE #kw_01_01
  LMOVE kw_01_02
  CLOSEI
  TWAIT 0.5
  LMOVE kw_01_01
  JMOVE kw_01_03
  LMOVE kw_01_04
  SPEED 1 ALWAYS
  OPENI
  TWAIT 0.5
  LMOVE kw_01_05
  TWAIT 1
  LMOVE kw_01_06
  TWAIT 1
  SPEED 100 ALWAYS
  LMOVE kw_01_03
  JMOVE #kw_02_01
  LMOVE kw_02_02
  CLOSEI
  TWAIT 0.5
  LMOVE kw_02_01
  JMOVE kw_02_03
  LMOVE kw_02_04
  SPEED 1 ALWAYS
  OPENI
  TWAIT 0.2
  LMOVE kw_02_05
  TWAIT 1
  LMOVE kw_02_06  TWAIT 1
  SPEED 100 ALWAYS
  LMOVE kw_02_03
  JMOVE #kw_03_01
  LMOVE kw_03_02
  CLOSEI
  TWAIT 0.5
  LMOVE kw_03_01
  JMOVE kw_03_03
  LMOVE kw_03_04
  SPEED 1 ALWAYS
  OPENI
  TWAIT 0.2
  LMOVE kw_03_05
  TWAIT 1
  LMOVE kw_03_06
  TWAIT 1
  SPEED 100 ALWAYS
```

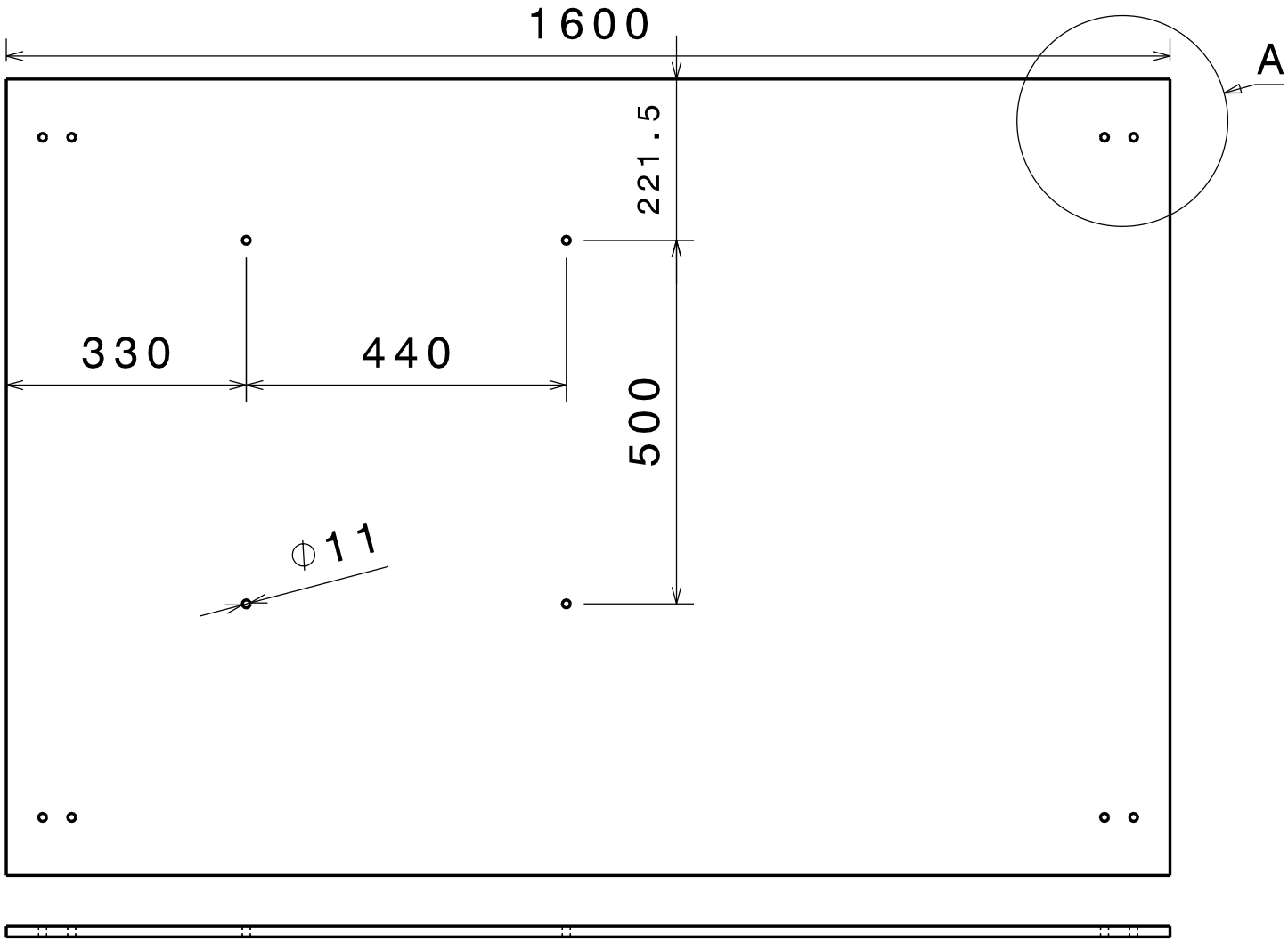
```
  LMOVE kw_03_03
  JMOVE #kw_04_01
  LMOVE kw_04_02
  CLOSEI
  TWAIT 0.5
  LMOVE kw_04_01
  JMOVE kw_04_03
  LMOVE kw_04_04
  SPEED 1 ALWAYS
  OPENI
  TWAIT 0.2
  LMOVE kw_04_05
  TWAIT 1
  LMOVE kw_04_06
  TWAIT 1
  SPEED 100 ALWAYS
  LMOVE kw_04_03
  kw_st1=1
.END


.PROGRAM kw_part1()
  SPEED 100 ALWAYS
  JMOVE #kw_05_01
  JMOVE kw_05_02
  SPEED 30 ALWAYS
  LMOVE kw_05_03
  CLOSEI
  TWAIT 1
  SPEED 1 ALWAYS
  LMOVE kw_05_04
  SPEED 30 ALWAYS
  LMOVE kw_05_05
  JMOVE kw_05_06
  SPEED 0.5
  LMOVE kw_05_07
  SPEED 0.5 ALWAYS
  LMOVE kw_05_08
  TWAIT 0.5
  LMOVE kw_05_09
  OPENI
  TWAIT 0.5
  SPEED 10
  DRAW 0,0,50
  SPEED 100
  DRAW 0,0,150
  kw_st2 = 1
.END


.PROGRAM kw_elastic()
  SPEED 100 ALWAYS
```

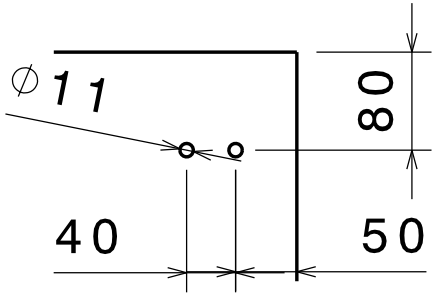
```
  JMOVE kw_06_02
  SPEED 40 ALWAYS
  LMOVE kw_06_03
  LMOVE kw_06_04
  CLOSEI
  TWAIT 1
  LMOVE kw_06_05
  SPEED
  JMOVE kw_06_07
  LMOVE kw_06_08  JMOVE kw_06_09
  SPEED 20
  JMOVE kw_06_10
  LMOVE kw_06_11
  LMOVE kw_06_12
  JMOVE kw_06_13
  SPEED 1 ALWAYS
  LMOVE kw_06_14
  OPENI
  TWAIT 0.5
  SPEED 30 ALWAYS
  DRAW 0,0,40
  SPEED 100 ALWAYS
  DRAW 0,0,60
  JMOVE kw_06_15
  SPEED 5 ALWAYS
  JMOVE kw_06_16
  TWAIT 0.5
  DRAW 0,0,100
  kw_st3 = 1
.END


.PROGRAM kw_screw1()
  SPEED 30 ALWAYS
  JMOVE #kw_07_01
  LMOVE kw_07_02
  SPEED 1 ALWAYS
  TWAIT 0.5
  CLOSEI
  DRAW 0,0,-4,0,0,-120,0.5
  DRAW 0,0,-12,0,0,0,0.1
  TWAIT 1
  DRAW 0,0,10,0,0,0,0.1
  SPEED 30
  LMOVE kw_07_04
  SPEED 100 ALWAYS
  JMOVE kw_07_05
  DRAW 0,0,-50,0,0,0,50
  TWAIT 0.5
  SPEED 0.5
  LMOVE kw_07_06
  OPENI
  DRAW 3,3,0,0,0,0,0.5
```

```
  DRAW -3,-3,0,0,0,0,0.5
  TWAIT 0.5
  TWAIT 1
  DRAW 0,0,30
  SPEED 10 ALWAYS
  TWAIT 0.5
  LMOVE kw_07_08
  SPEED 1 ALWAYS
  LMOVE kw_07_09
  TWAIT 0.5
  LMOVE kw_07_08
  LMOVE kw_07_10
  TWAIT 0.5
  SPEED 0.1 ALWAYS
  DRAW 0,0,-4,0,0,-20,0.1
  TWAIT 0.5
  DRAW 0,0,-1,0,0,0,0.1
  DRIVE 6,360,5
  DRIVE 6,180,5
  TWAIT 0.5
  SPEED 5 ALWAYS
  DRAW 0,0,50
  SPEED 100 ALWAYS
  DRAW 0,0,150
  kw_st4 = 1
.END


.PROGRAM kw_screw2()
  SPEED 30 ALWAYS
  JMOVE #kw_08_01
  LMOVE kw_08_02
  SPEED 1 ALWAYS
  TWAIT 0.5
  CLOSEI
  DRAW 0,0,-5,0,0,-120,0.5
  DRAW 0,0,-9,0,0,0,0.1
  TWAIT 1
  DRAW 0,0,10,0,0,0,0.1
  SPEED 30
  LMOVE kw_08_04
  SPEED 100 ALWAYS
  JMOVE kw_08_05
  DRAW 0,0,-50,0,0,0,50
  TWAIT 0.5
  SPEED 0.5
  LMOVE kw_08_06
  OPENI
  DRAW -3,-3,0,0,0,0,0.5
  DRAW 3,3,0,0,0,0,0.5
  TWAIT 0.5
  TWAIT 1
  DRAW 0,0,30
```

```
  SPEED 10 ALWAYS
  TWAIT 0.5
  LMOVE kw_08_08
  SPEED 1 ALWAYS
  LMOVE kw_08_09
  TWAIT 0.5
  LMOVE kw_08_08
  LMOVE kw_08_10
  TWAIT 0.5
  SPEED 0.1 ALWAYS
  DRAW 0,0,-4,0,0,-20,0.1
  DRAW 0,0,-1,0,0,0,0.1
  TWAIT 0.5
  DRIVE 6,360,5
  DRIVE 6,220,5
  TWAIT 0.5
  SPEED 5 ALWAYS
  DRAW 0,0,50
  SPEED 100 ALWAYS
  DRAW 0,0,150
  kw_st5 = 1
.END
```

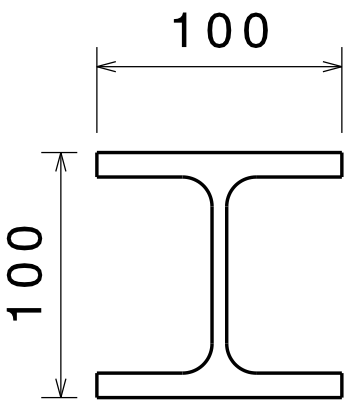
9. ANNEX 2: PLANS

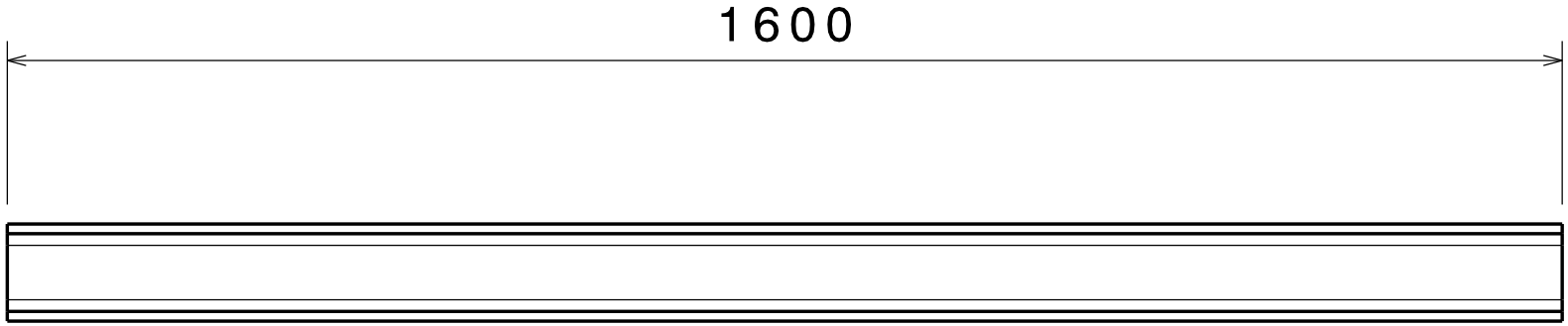




Detail A
Scale 1:8

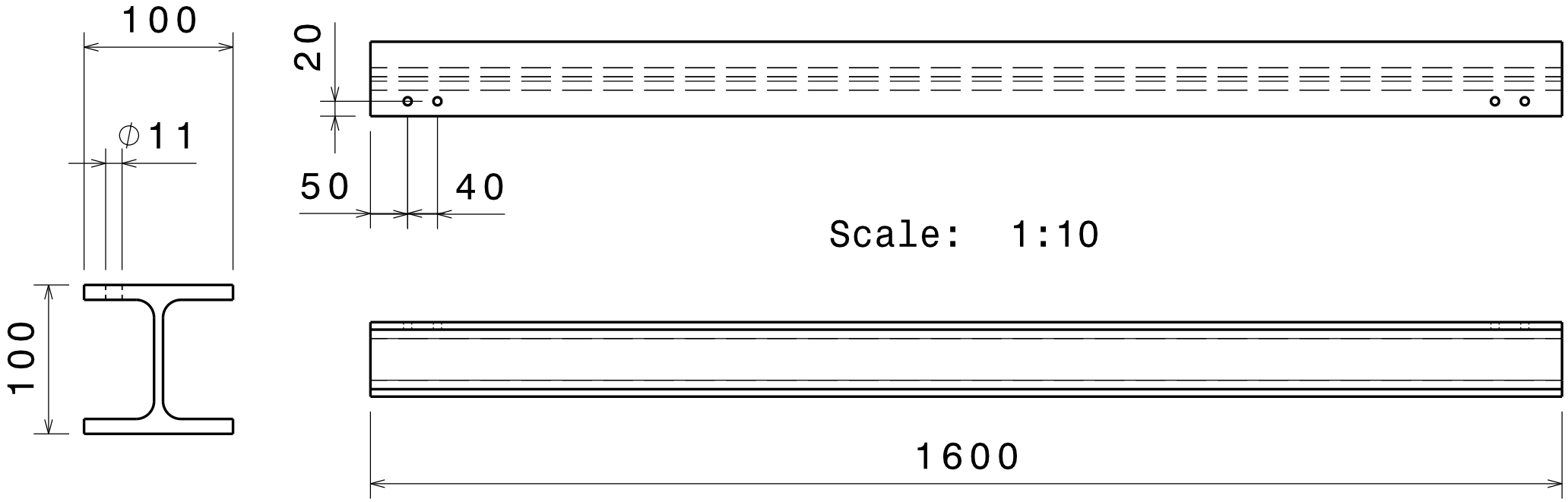
HOCHSCHULE OSNABRÜCK	I. und I. INDUSTRIAL ENGINEERING	LABORATORY: LABOR FÜR HANDHABUNGSTECHNIK UND ROBOTIK
MASTER´S THESIS		PERFORMED BY: TROYAS GARCÍA, GONZALO SIGN.:
PLAN: PLATFORM - PLATE		DATE: 06.05.2013 / SCALE: 1:12 / PLAN #: 3.1.



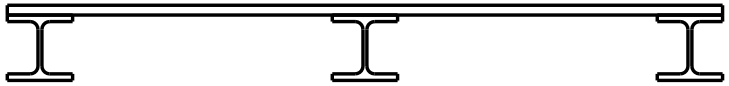


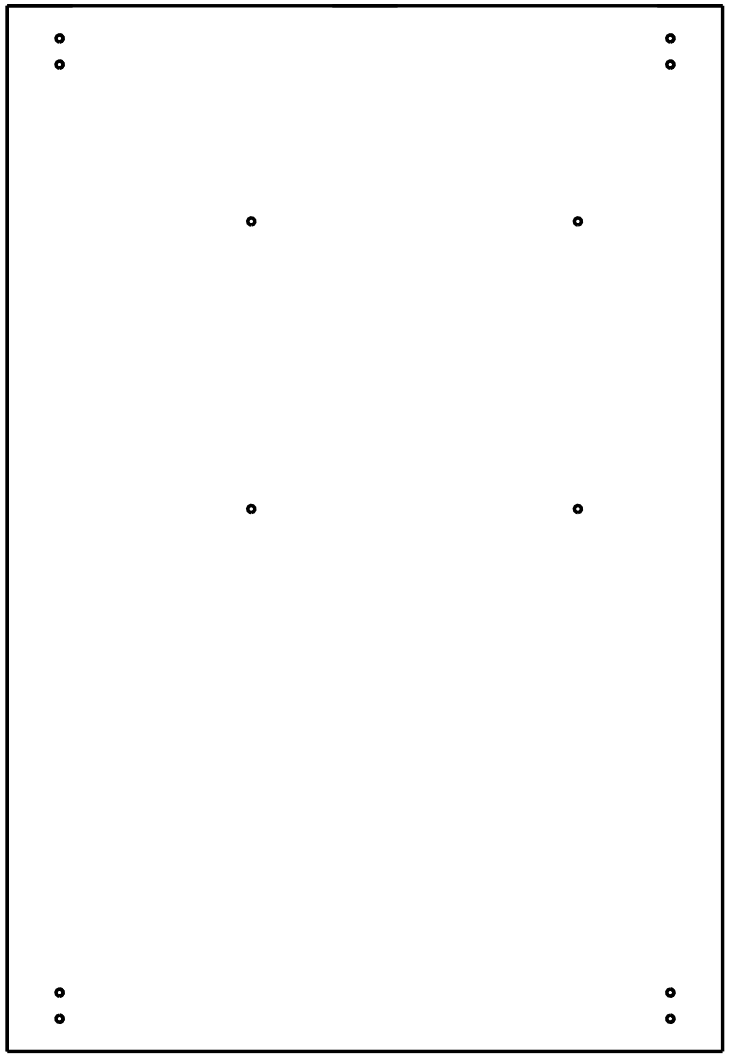
Scale: 1:10

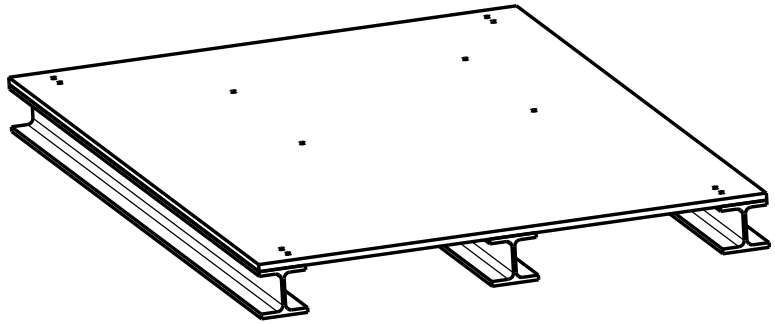
HOCHSCHULE OSNABRÜCK	I. und I. INDUSTRIAL ENGINEERING	LABORATORY: LABOR FÜR HANDHABUNGSTECHNIK UND ROBOTIK
MASTER´S THESIS		PERFORMED BY: TROYAS GARCÍA, GONZALO SIGN.:
PLAN: PLATFORM - DOUBLE-T PROFILE - CENTER	DATE: 06.05.2013 / SCALE: 1:5	PLAN #: 3.2.



HOCHSCHULE OSNABRÜCK	I. und I. INDUSTRIAL ENGINEERING	LABORATORY: LABOR FÜR HANDHABUNGSTECHNIK UND ROBOTIK
MASTER´S THESIS		PERFORMED BY: TROYAS GARCÍA, GONZALO SIGN.:
PLAN: PLATFORM - DOUBLE-T - SIDES	DATE: 06.05.2013	SCALE: 1:5 / PLAN #: 3.3.

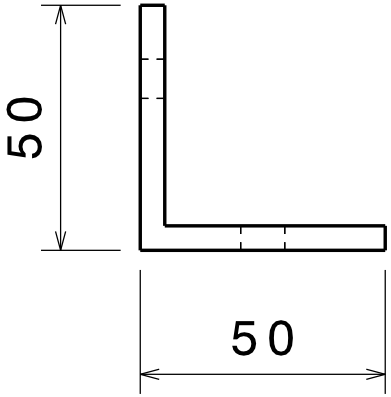


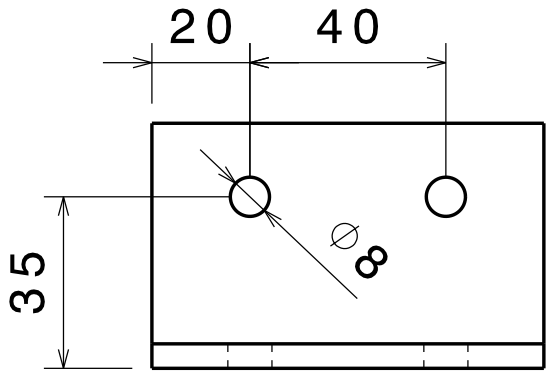


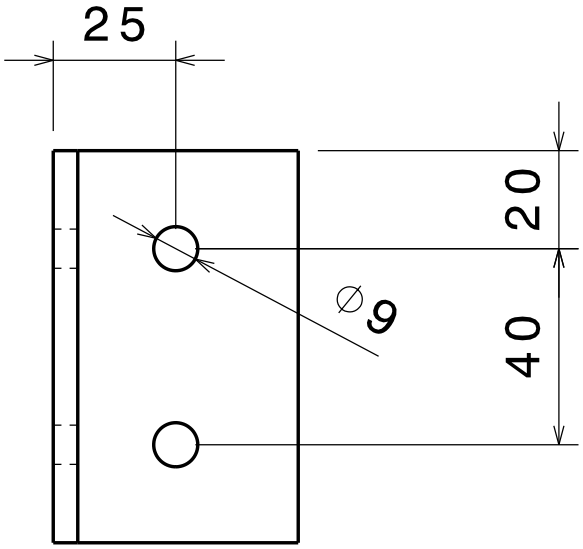


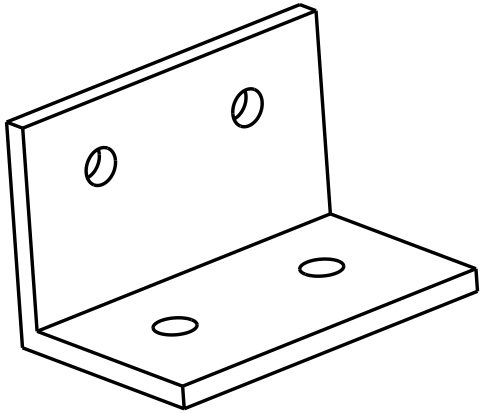
Scale: 1:20

HOCHSCHULE OSNABRÜCK	I. und I. INDUSTRIAL ENGINEERING	LABORATORY: LABOR FÜR HANDHABUNGSTECHNIK UND ROBOTIK
MASTER´S THESIS		PERFORMED BY: TROYAS GARCÍA, GONZALO
		SIGN.:
PLAN: PLATFORM	DATE: 06.05.2013	SCALE: 1:15
		PLAN #: 3.4.

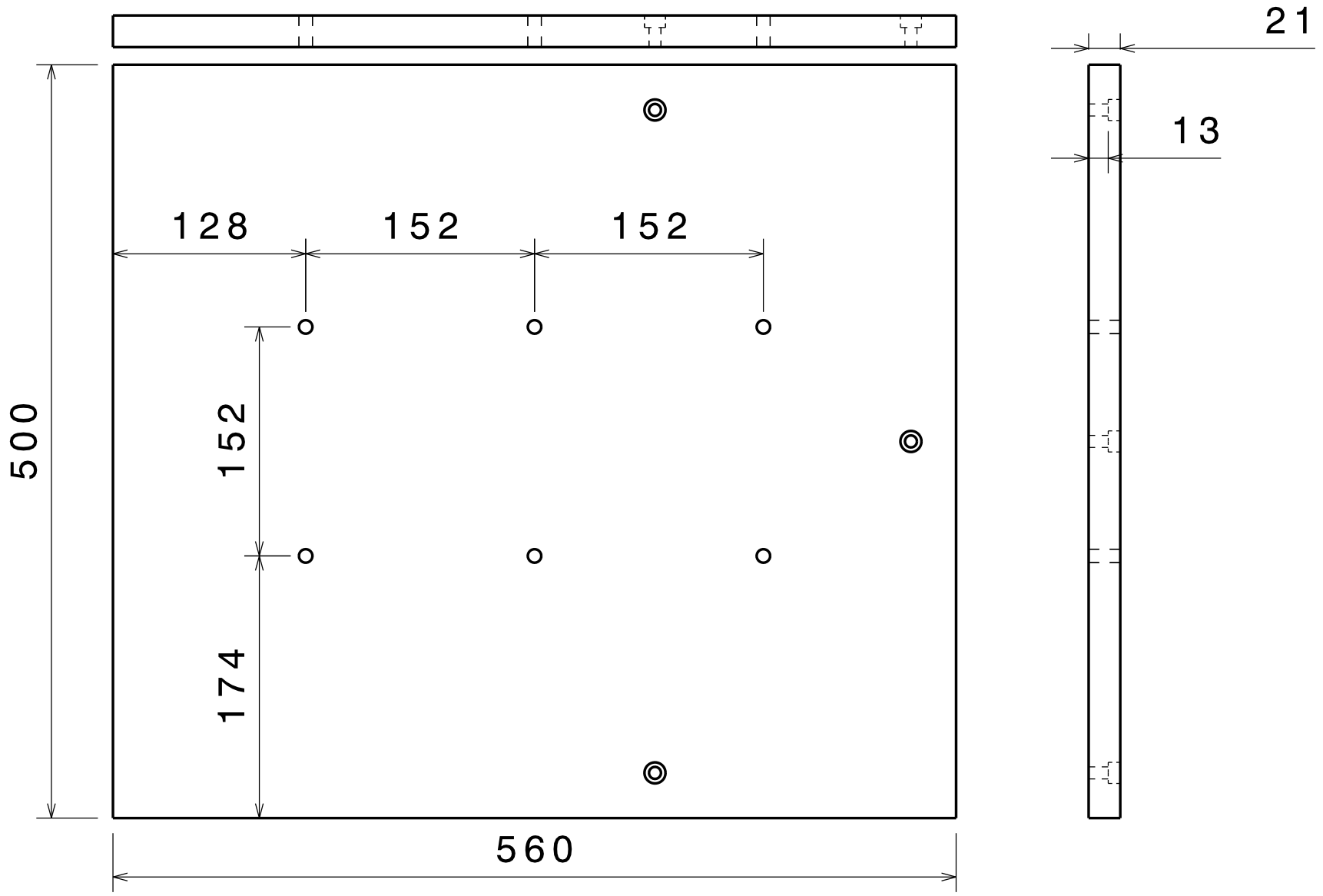




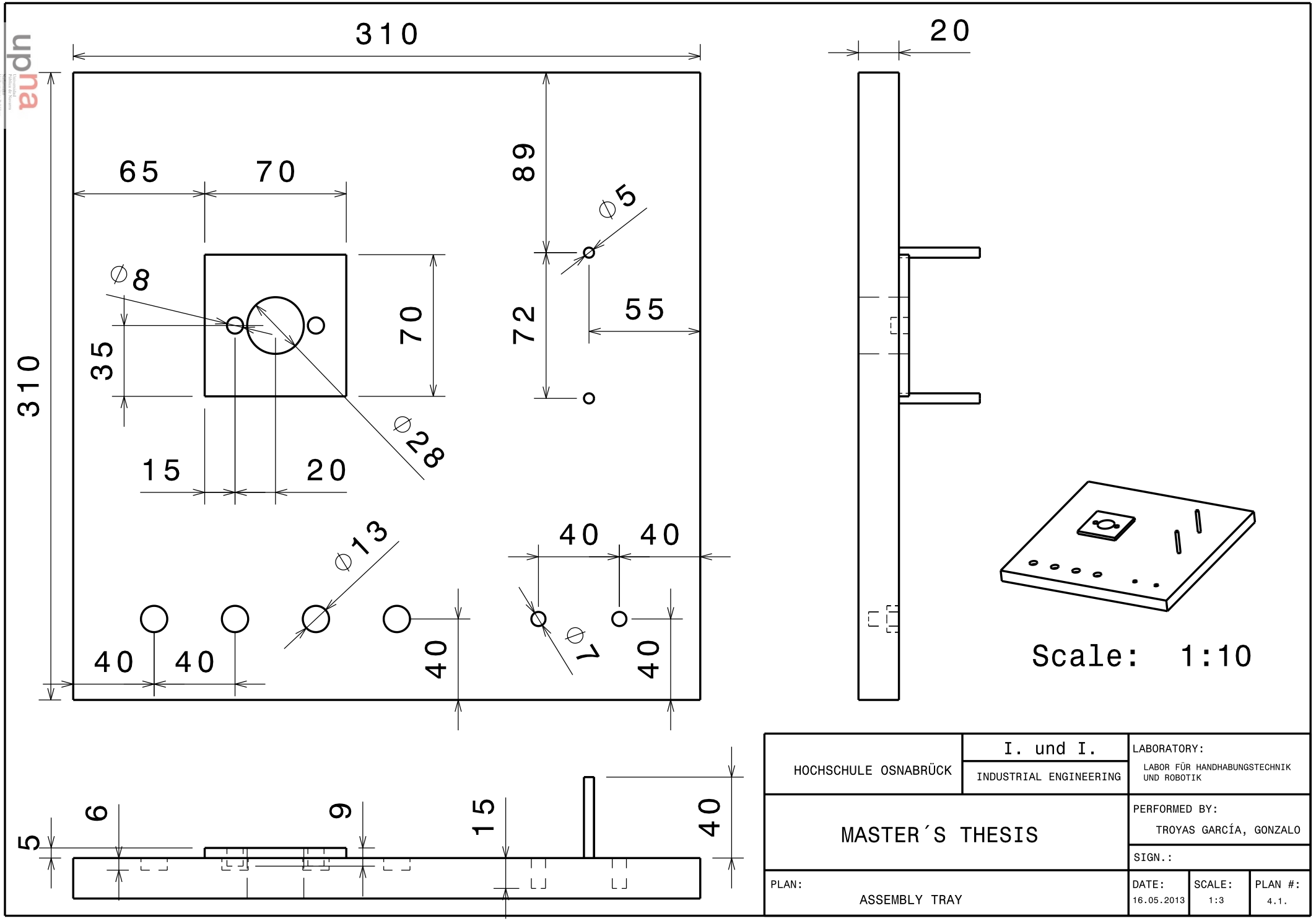




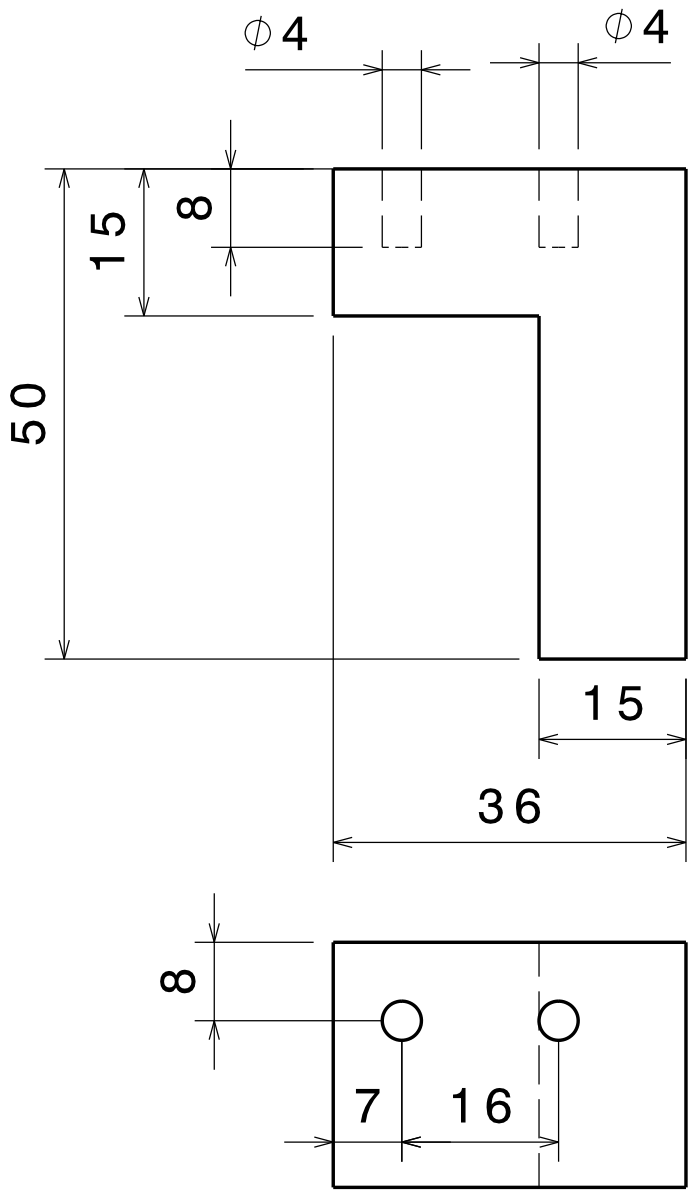
HOCHSCHULE OSNABRÜCK	I. und I. INDUSTRIAL ENGINEERING	LABORATORY: LABOR FÜR HANDHABUNGSTECHNIK UND ROBOTIK
MASTER´S THESIS		PERFORMED BY: TROYAS GARCÍA, GONZALO SIGN.:
PLAN: PLATFORM - L-PROFILE		DATE: 06.05.2013 / SCALE: 1:2 / PLAN #: 3.5.

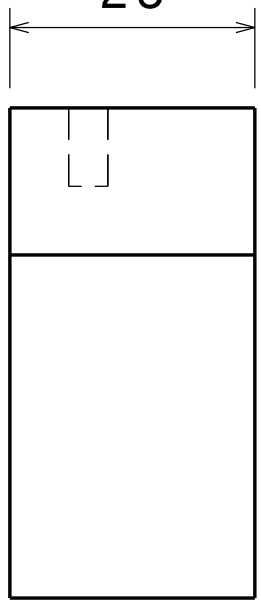


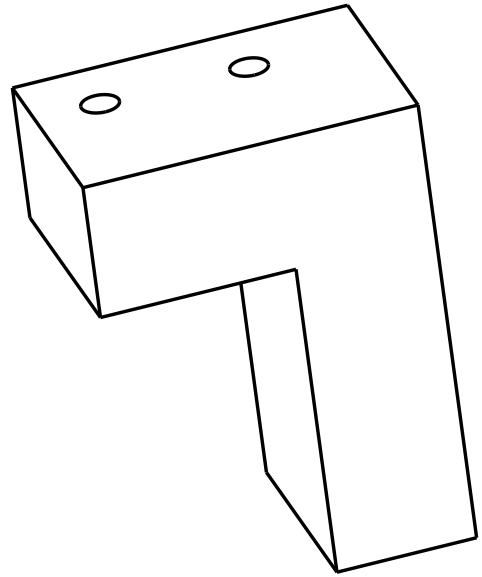
HOCHSCHULE OSNABRÜCK	I. und I. INDUSTRIAL ENGINEERING	LABORATORY: LABOR FÜR HANDHABUNGSTECHNIK UND ROBOTIK
MASTER´S THESIS		PERFORMED BY: TROYAS GARCÍA, GONZALO SIGN.:
PLAN: PLATE OF THE KAWASAKI´S STRUCTURE		DATE: 07.05.2013 / SCALE: 1:5 / PLAN #: 3.6.



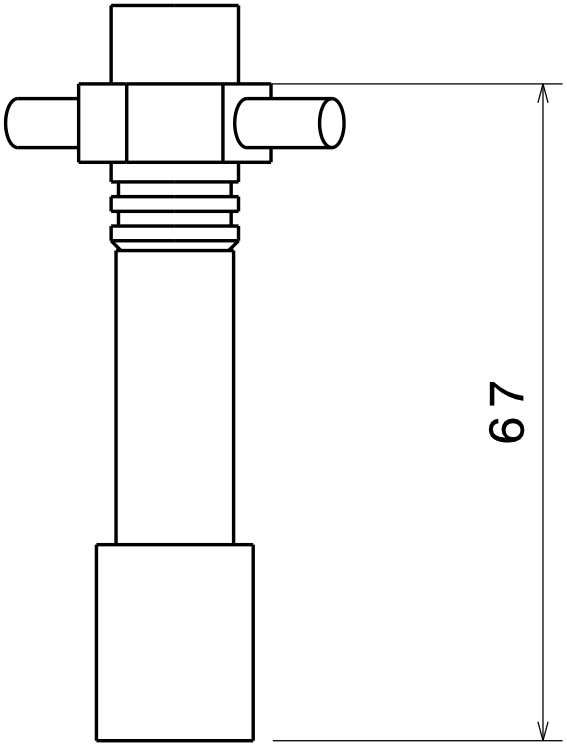
HOCHSCHULE OSNABRÜCK	I. und I. INDUSTRIAL ENGINEERING	LABORATORY: LABOR FÜR HANDHABUNGSTECHNIK UND ROBOTIK	
MASTER´S THESIS		PERFORMED BY: TROYAS GARCÍA, GONZALO SIGN.:	
PLAN: ASSEMBLY TRAY		DATE: 16.05.2013	SCALE: 1:3	PLAN #: 4.1.

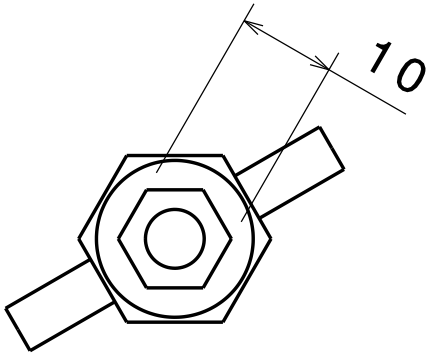


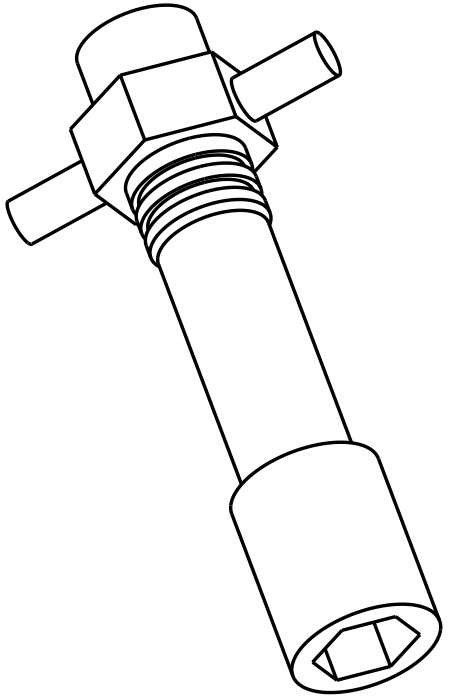




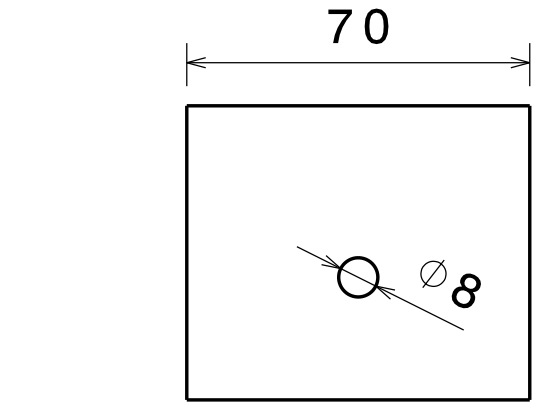
HOCHSCHULE OSNABRÜCK	I. und I. INDUSTRIAL ENGINEERING	LABORATORY: LABOR FÜR HANDHABUNGSTECHNIK UND ROBOTIK
MASTER´S THESIS		PERFORMED BY: TROYAS GARCÍA, GONZALO SIGN.:
PLAN: GRIPPER FINGERS	DATE: 10.05.2013	SCALE: 1:1
		PLAN #: 4.2.

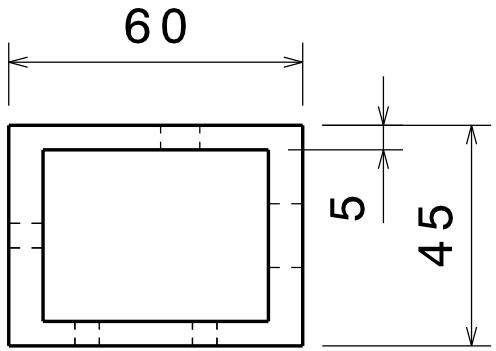


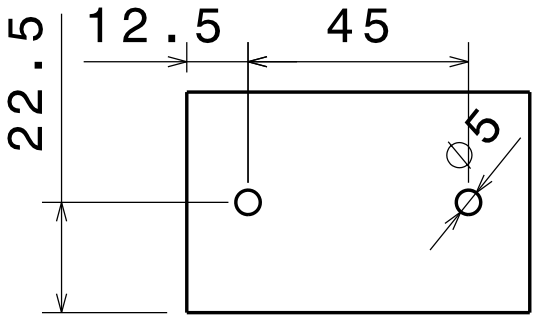


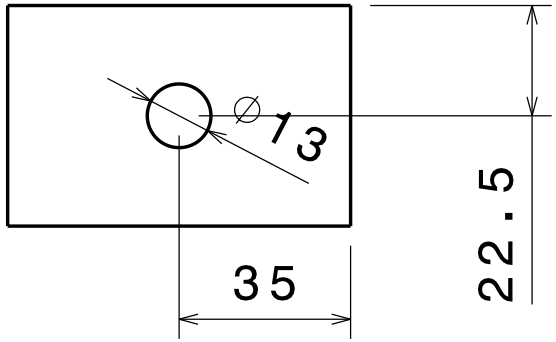


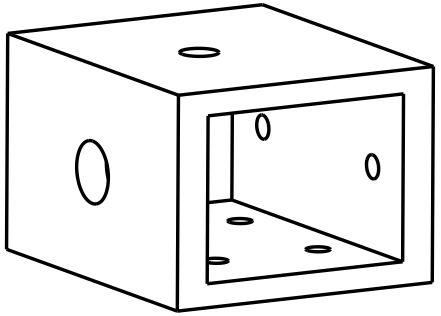
HOCHSCHULE OSNABRÜCK	I. und I. INDUSTRIAL ENGINEERING	LABORATORY: LABOR FÜR HANDHABUNGSTECHNIK UND ROBOTIK
MASTER´S THESIS		PERFORMED BY: TROYAS GARCÍA, GONZALO SIGN.:
PLAN: SCREWING TOOL		DATE: 10.05.2013 / SCALE: 1:1 / PLAN #: 4.3.

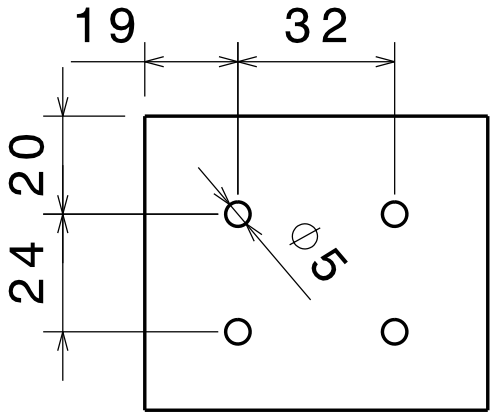












HOCHSCHULE OSNABRÜCK	I. und I. INDUSTRIAL ENGINEERING	LABORATORY: LABOR FÜR HANDHABUNGSTECHNIK UND ROBOTIK
MASTER´S THESIS		PERFORMED BY: TROYAS GARCÍA, GONZALO SIGN.:
PLAN: TOOLS ADAPTER	DATE: 10.05.2013	SCALE: 1:1
		PLAN #: 4.4.